Análisis de Sistemas Operativos de Tiempo Real Libres

© 2004 Diego LÓPEZ ZAMARRÓN

Copyright (c) 2004 Diego LÓPEZ ZAMARRÓN
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Tabla de Contenidos

1.- Introducción....4
2.- Arquitectura de un Sistema Operativo de Tiempo Real....5
2.1.- Arquitectura de un Sistema Operativo de Propósito General....5
2.2.- Clases de tiempo real....6
2.3.- Características de rendimiento....6
2.4.- Arquitectura de un Sistema Operativo de Tiempo Real....8
2.5.- Rendimiento de un sistema....12
2.6.- Enlaces....13
3.- Distribuciones....14
3.1.- Introducción....14
3.2.- Clasificación....15
3.3.- Descripción....16
3.3.1.- ADEOS – Adaptative Domain Environment Operating Systems....16
3.3.2.- RTAI – Real Time Application Interface....19
3.3.3.- RTLinux....24
3.3.4.- ART Linux....27
3.3.5.- KURT – Kansas University Real Time....28
3.3.6.- Linux/RK – Linux/Resource Kernel....30
3.3.7.- Qlinux....31
3.3.8.- RED-Linux....32
3.3.9.- BlueCat RT....33
3.3.10.- RedHawk....35
3.3.11.- REDICE-Linux....36
3.3.12.- TimeSys Linux....38
3.3.13.- Linux-SRT....39
4.- Elección de distribución/es....40
4.1.- Manuales de Instalación....40
4.1.1.- Instalación de ADEOS....40
4.1.2.- Instalación de RTAI sobre ADEOS....42
4.2.- Enlaces....44
5.- Driver sobre la distribución elegida....45
5.1.- Descripción del hardware....45
5.1.1.- Introducción....45
5.1.2.- Descripción del puerto paralelo....45
5.1.3.- Descripción de la placa....46
5.2.- Descripción del software....52
5.2.1.- Driver....53
5.2.2.- Librería....54
5.2.3.- Aplicación....55
Anexo 1: GNU Free Documentation License....56

© 2004 Diego LÓPEZ ZAMARRÓN

1.- Introducción

El presente trabajo tiene como objetivo principal proporcionar un resumen del conocimiento sobre el diseño y las posibles aplicaciones de los sistemas operativos de tiempo real en general y de sus distintas distribuciones en particular.

Dentro de las distribuciones de sistemas operativos de tiempo real, nos fijaremos en aquellas que usan como base el kernel estándar de Linux, en sus distintas versiones, y más en particular en aquellas que se distribuyen bajo la licencia GNU/GPL ya que de estas últimas obtendremos mucha mayor información y será una de estas distribuciones libres las que escogamos para nuestro trabajo.

En primer lugar, se realizará una descripción de las características principales requeridas para un sistema operativo de tiempo real y las distintas arquitecturas de sistemas operativos que se han adoptado para mejorar las características y convertir un sistema operativo de propósito general en uno de tiempo real. Se mostrará un tabla resumen del rendimiento de cada una de las distintas arquitecturas.

En segundo lugar, se hará una introducción a distintas distribuciones de sistemas operativos de tiempo real, realizando una pequeña clasificación y comentarios acerca de su utilidad en distintos ambientes. Destacaremos también el tipo de licencia bajo la cual se distribuyen.

En tercer lugar, se elegirá una de las distribuciones y se procederá a instalar, por lo que se desarrollará un manual de instalación.

Y en cuarto lugar, se realizará un driver, sobre la distribución elegida, de una tarjeta de 32 entradas y 32 salidas digitales controlada a través del puerto paralelo. Esta tarjeta ha sido desarrollada por el departamento DISAM (www.disam.upm.es) de la UPM (www.upm.es) dentro del proyecto PAUTA. Se realizará tambien una aplicación de usuario que haga uso del driver.

La última versión de este documento se encuentra disponible a través de:

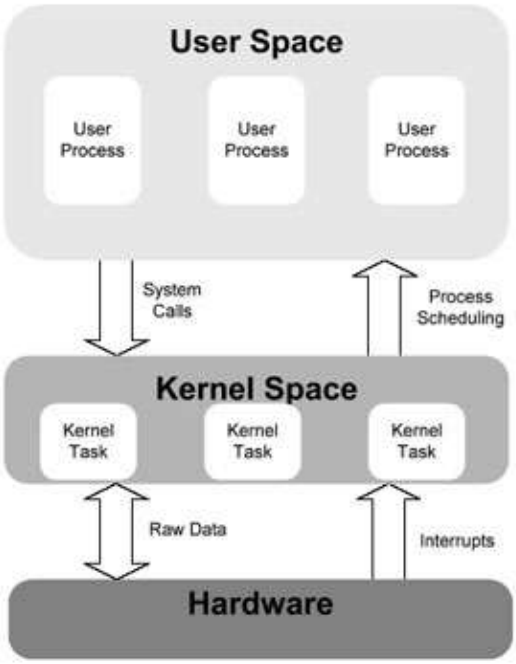
- Formato PDF: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/sotr.pdf
- Formato OpenOffice: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/sotr.sxw

© 2004 Diego LÓPEZ ZAMARRÓN

2.- Arquitectura de un Sistema Operativo de Tiempo Real

2.1.- Arquitectura de un Sistema Operativo de Propósito General

La memoria física de un computar esta dividida entre el espacio reservado para los usuarios (“user-space”) y el espacio reservado para el kernel (“kernel-space”). El kernel multitarea es capaz de manejar múltiples aplicaciones de usuarios que ejecutan en el espacio de usuario haciendo creer a cada uno que dispone de todo el espacio de memoria y de todos los recursos hardware. La comunicación entre los programas en el espacio de usuario y el espacio del kernel se realiza a través de las llamadas al sistema. Estas llamadas típicamente lo que hacen es acceder a recursos físicos compartidos. Todos los accesos a los recursos hardware son controlados por el kernel de modo que los programas de usuario no conozcan los detalles físicos de los dispositivos.



Las funcionalidades principales de un sistema operativo de propósito general son:

- Gestión de procesos. Planificación de procesos.
- Gestión de memoria.
- Interactuar con el Hardware.
- Servidor de Ficheros.
- Servidor de Comunicaciones.

© 2004 Diego LÓPEZ ZAMARRÓN

La funcionalidad principal y requerida para un sistema operativo de tiempo real es proveer de un nivel de servicio adecuado a las aplicaciones que requieran una respuesta en un intervalo de tiempo determinado.

La característica principal de un sistema operativo de tiempo real es la respuesta ante eventos internos ó externos, tales como interrupciones hardware externas, interrupciones software internas ó interrupciones de reloj internas, es decir los requerimientos temporales.

Una de las medidas de rendimiento de un Sistema Operativo de Tiempo Real es la latencia, ó tiempo desde que ocurre el evento y éste es tratado. La otra medida es el jitter, ó variaciones en el periodo normal de ocurrencia de eventos periódicos.

Todos los sistemas operativos tienden a tener una baja latencia y un bajo jitter, pero los sistemas operativos de tiempo real requieren que esos valores estén determinados y que no dependan de la carga del sistema.

2.2.- Clases de tiempo real

Un programa ó un sistema operativo es considerado como de tiempo real, si a pesar de las restricciones de tiempo le permiten trabajar y funcionar correctamente.

Se distinguen las siguientes clases:

- Tiempo real estricto (Hard Real Time): Todas las acciones deben ocurrir dentro del plazo especificado.
- Tiempo real flexible (Soft Real Time): Se pueden perder plazos de vez en cuando. El valor de la respuesta decrece con el tiempo.
- Tiempo real firme (Firm Real Time): Se pueden perder plazos ocasionalmente. Una respuesta tardía no tiene valor.

2.3.- Características de rendimiento

El criterio fundamental de evaluación del rendimiento de un sistema operativo de tiempo real es la latencia y el periodo del jitter ante un evento. Un evento es cualquier tipo de interrupción, tanto interna, como externa.

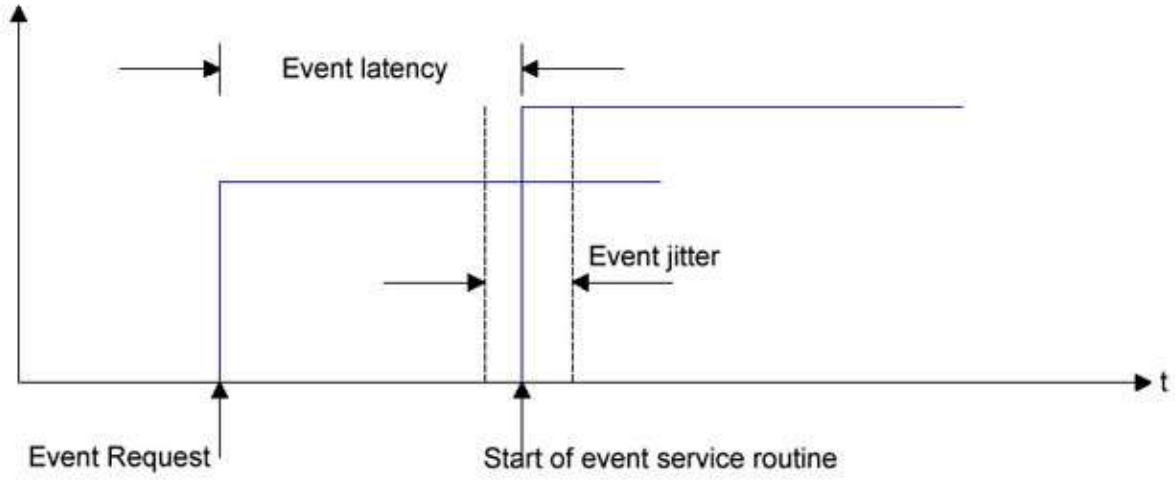
- Latencia en un evento

  Un evento puede ser tanto una interrupción hardware como una interrupción software.

  La latencia ante una interrupción hardware es el tiempo desde que se produce la interrupción hasta que se ejecuta la primera instrucción de la rutina de tratamiento. Puede haber retrasos debido al acceso al bus.

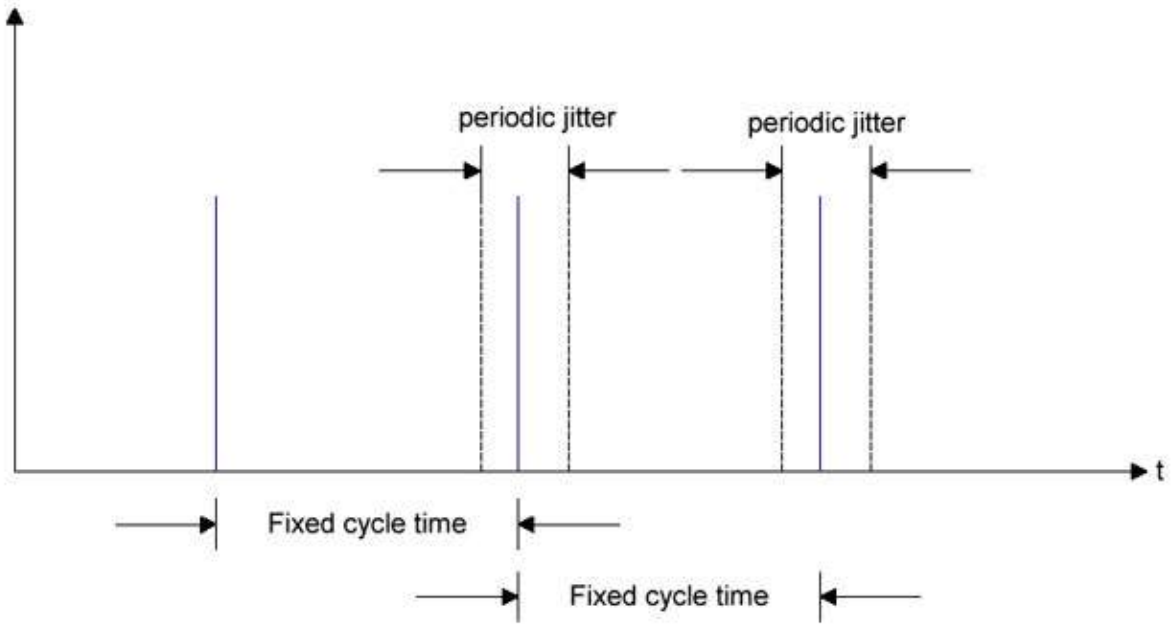
  La latencia ante una interrupción software es el tiempo desde que la señal es generada hasta que la primera instrucción de la tarea es ejecutada. Aquí el valor depende únicamente del acceso a los registros del procesador.

© 2004 Diego LÓPEZ ZAMARRÓN



- Periodo del Jitter

  El periodo del jitter se refiere a las variaciones en el tiempo que experimenta una tarea cuando se ejecuta de manera repetitiva.



2.4.- Arquitectura de un Sistema Operativo de Tiempo Real

El objetivo de un sistema operativo de tiempo real es reducir la latencia y el jitter en las interrupciones, tanto internas como externas, al orden de microsegundos.

Es decir, la parte fundamental para convertir un sistema operativo de propósito general en un sistema operativo de tiempo real es el manejo de las interrupciones.

El procesamiento de interrupciones en el kernel estándar esta divido en 2 tareas. Una tarea que se encarga de leer los datos del dispositivo físico y escribirlos en un buffer, es lo que se conoce como manejador de interrupciones, y una tarea que se encarga de pasar los datos del buffer a otro para que sean accesible por el kernel. Con este esquema, cuando el manejador esta ejecutando, todas las interrupciones están inhibidas con el siguiente retardo impredecible en el servicio de otras interrupciones que se puedan haber producido y por tanto en los valores de latencia y jitter.

Para conseguir reducir la latencia y el jitter se han desarrollado distintas alternativas que modifican el kernel de linux en este aspecto fundamentalmente.

Actualmente hay dos corrientes de diseño:

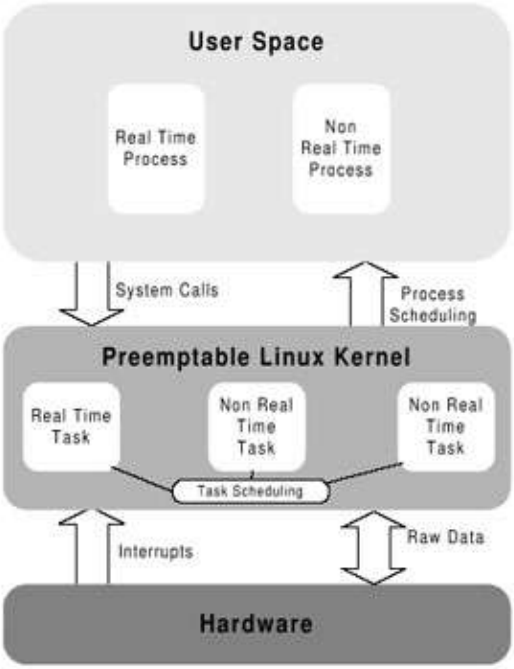
- *Atención prioritaria en el kernel estándar (Preemptable kernel)*

  Esta metodología modifica el kernel en profundidad de forma que los procesos de kernel ejecuten con máxima prioridad de forma que puedan interrumpir a procesos de menor prioridad en el acceso a los recursos que necesiten.

  Esta metodología implica cambios en los manejadores de interrupciones para que las interrupciones de alta prioridad no sean bloqueadas por el manejador de interrupciones mientras esta manejando otra de menor prioridad.

  El resultado de esta metodología es una latencia y un jitter del orden de 1 milisegundo en un Pentium a 100 Mhz.

  A partir de la versión 2.5.4 del kernel de Linux se incorpora esta metodología.



Como se puede observar en la figura la tarea de tiempo real esta controlada por el planificador del kernel y es una más de las tareas que controla el kernel. Esta tarea hace referencia a los procesos de tiempo real en el espacio de usuario. El planificador sabe que las tareas de tiempo real tiene mayor prioridad que las tareas que no son de tiempo real.

Como se puede comprobar esta metodología es adecuada para aplicaciones de audio y vídeo donde el periodo de interrupciones es del orden de 1 milisegundo, pero inadecuado cuando ya hablamos de menos de 1 milisegundo.

Beneficios y limitaciones de la estrategia de kernel preemptable:

Beneficios	***Limitaciones***
Desarrollo de aplicaciones de tiempo real más fácil	El rendimiento no es lo suficientemente bueno para los requerimientos de baja latencia en el rango de microsegundos
Protección de memoria disponible	El peor caso de latencia de una interrupción es desconocido ya que no es posible testearlo
Los programas de tiempo real no pueden quebrar el kernel	Cambio fuerte en el código fuente del kernel
Acceso completo a los servicios del sistema (TCP/IP, I/O)	Cada vez que sale una nueva versión del kernel es necesario un test profundo
Threads de POSIX disponibles para las funciones de tiempo real	Para realizar el análisis de funcionamiento son necesarios todos los drivers de dispositivos y módulos
Soporte de herramientas para facilitar la depuración	Rendimiento global del sistema reducido

- *Modificaciones sobre el kernel estándar (Patch)*

Existen 4 estrategias de modificación del kernel de Linux para proveer capacidades de tiempo real. Tres de ellas implican añadir un segundo kernel (kernel dual) para manejar las tareas de tiempo real y el cuarto implica modificar directamente el código del kernel para añadir características de tiempo real.

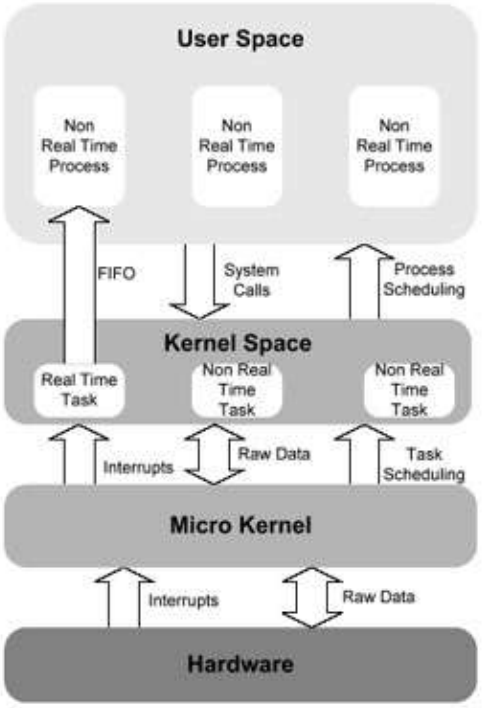
Beneficios y limitaciones de la estrategia de kernel dual:

Beneficios	***Limitaciones***
Cambios de contexto y latencia de interrupciones muy bajo (5 – 10 microsegundos)	Todas las características de tiempo real deben ser implementadas como módulos
El kernel estándar de Linux ejecuta como una tarea de baja prioridad del microkernel	El desarrollo de aplicaciones de tiempo real es mucho más complicado
Capa de abstracción hardware e interrupciones	Se debe ser conocedor de Linux en profundidad
Garantizada una planificación determinística	Interacción entre el kernel y los drivers de los dispositivos
	Código incorrecto puede provocar el fallo del kernel
	Más difícil depurar el código
	Las llamadas al sistemas de Linux no pueden tomar el control y el rendimiento no puede ser garantizado

© 2004 Diego LÓPEZ ZAMARRÓN

- **Micro-kernel**

  Esta estrategia añade un segundo kernel que en realidad es una capa interfaz entre el hardware y el kernel estándar, lo que se llama tradicionalmente HAL "Hardware Abstraction Layer". Esta capa, micro-kernel, controla la ejecución de las tareas de tiempo real y ejecuta el kernel estándar como una tarea en background, es decir, el kernel estándar sólo ejecuta cuando no hay tareas de tiempo real pendientes.



  Este microkernel intercepta las interrupciones hardware y asegura que las tareas de tiempo real ejecuten con la mayor prioridad posible de forma que la latencia se minimice.

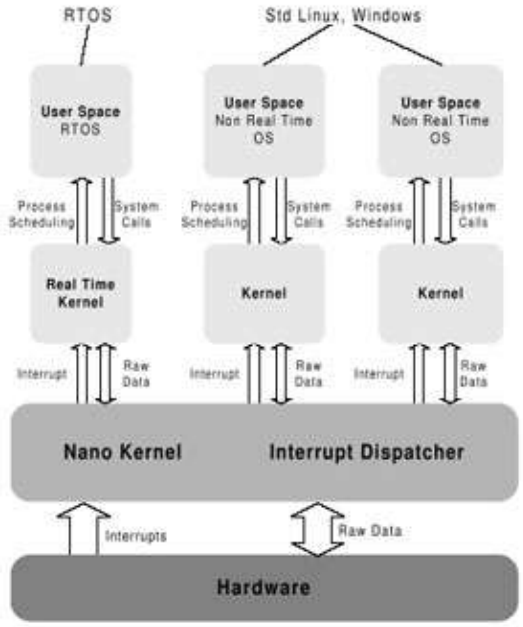
  Ejemplo de implementación de esta metodología son RTLinux y RTAI.

- **Nano-kernel**

  Esta estrategia es similar a la primera, micro-kernel, pero consigue evitar la patente que tiene Yodaiken sobre ésta, de forma que este nano-kernel únicamente captura las interrupciones hardware y permite la ejecución paralela de varios sistemas operativos por encima de él.

  En este sentido, esta estrategia no desemboca directamente en un sistema operativo de tiempo real, sino que simplemente es una capa intermedia entre el hardware y un sistema operativo, que puede ser de tiempo real ó no.

© 2004 Diego LÓPEZ ZAMARRÓN

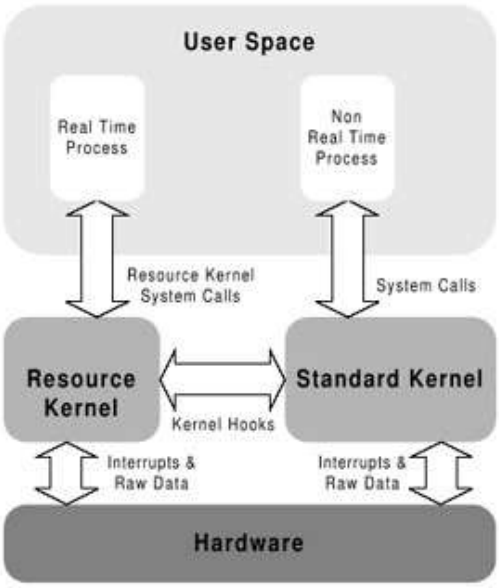


Una implementación de esta estrategia es ADEOS. Los desarrolladores de este proyecto fueron precisamente los que pusieron el nombre de nano-kernel para diferenciarlo claramente de la estrategia de micro-kernel patentado por Yodaiken.

- **Extensión con un nuevo kernel de acceso a los recursos (Recurso-kernel)**

  Esta estrategia añade un kernel de forma que éste proporciona una puerta de acceso a los recursos, tales como al sistema de ficheros, al puerto paralelo, etc, tanto para el kernel estándar como para los procesos de usuario.

  El recurso kernel no sólo captura las interrupciones sino que proporciona un mecanismo donde los programas de usuario pueden requerir, reservar y garantizarse un porcentaje finito de los recursos como pueden ser de CPU, memoria, etc.



- **Extensiones POSIX de tiempo real añadidas al kernel**

  Esta estrategia consiste en modificar directamente el kernel estándar de Linux para añadir librerías que implementan las extensiones de tiempo real de POSIX. El resultado es un kernel conforme al estándar IEEE 1003.1d. No añade un segundo kernel.

  Las modificaciones realizadas al kernel consisten en la implementación de relojes, señales, semáforos, memoria compartida, planificador por prioridades, etc según lo especificado en IEEE 1003.1d.

  Existen 2 aproximaciones diferentes para esta estrategia:

  - KURT (The Kansas University Real Time Linux) que únicamente implementa los relojes conforme al estándar IEEE 1003.1d.
  - TimeSys Linux. Añade al "preemptable kernel" un planificador de kernel que proporciona una latencia y un jitter menor de 100µs. El parche con el planificador no proporciona una alta resolución en los relojes, que es necesaria para tareas de tiempo real repetitivas.

2.5.- Rendimiento de un sistema

Si comparamos por ejemplo el kernel estándar con RTLinux podremos observar claramente como existe gran diferencia tanto en la latencia en las interrupciones como en el jitter, siendo mucho menos para el caso de RTLinux y siempre en el orden de 1 a 10 microsegundos para un Pentium a 100Mhz.

Si comparamos entre si las distintas estrategias de diseño de un sistema operativo de tiempo real podremos observar que las diferencias no son tan amplias si las comparamos con el kernel estándar.

	Aplicación	**Latencia/Jitter**
estándar OS	Nom Real-Time	100µs to 100ms
estándar Linux (>2.5.X)	Soft Real-Time	1ms
IEEE 1003.1d	Hard Real-Time	10 to 100µs
Micro-Kernel	Hard Real-Time	1 to 10µs
RTOS Kernel	Hard Real-Time	1 to 10µs

© 2004 Diego LÓPEZ ZAMARRÓN

2.6.- Enlaces

RTOS: Real Time Operating Systems --> Sistemas Operativos de Tiempo Real

- RTOS General --> http://www.realtimelinuxfoundation.org/
- RTOS General --> http://www.linuxdevices.com/articles/AT8073314981.html
- RTOS General --> http://www.fact-index.com/r/re/real_time_operating_system.html
- RTOS General --> http://www.se.rit.edu/~jrv/research/RT_Embedded.html
- RTOS General --> http://www.linuxworld.com/story/34335.htm
- RTOS Arquitectura --> http://www.redsonic.com/products/real-time.htm
- RTOS Arquitectura --> http://linuxdevices.com/articles/AT4627965573.html
- RTOS Arquitectura --> http://www.linuxdevices.com/articles/AT4503827066.html
- RTOS Arquitectura --> http://www.linuxdevices.com/articles/AT7005360270.html
- RTOS Aplicación --> http://www.linuxdevices.com/articles/AT5709748392.html
- RTOS Análisis --> http://www.mnis.fr/opensource/ocera/rtos/
- Documentos:
  - EmbeddedLinux.ppt --> http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/embeddedlinux.ppt
  - Linux_as_RTOS.pdf --> http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/linux_as_rtos.pdf
  - RealTimeLinuxReport.pdf --> http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/realtimelinuxreport.pdf

© 2004 Diego LÓPEZ ZAMARRÓN

3.- Distribuciones

3.1.- Introducción

Buscando información sobre distribuciones de sistemas operativos de tiempo real he encontrado que hay alrededor de 30 distribuciones, la mayoría para el sistema operativo Linux y en menor medida para Windows, aunque en este trabajo nos hemos centrado en las disponibles bajo Linux y con más detalle en aquellas que son libres, y que se distribuyen bajo la licencia GNU/GPL.

De todas las que hay, he seleccionado una decena de ellas de las que se realizará una breve descripción de su arquitectura, su licencia y el entorno para la cual esta diseñada. Algunas de éstas están en continuo desarrollo y otras por el contrario están un poco estancadas ó definitivamente olvidadas, pero que son nombradas porque nacieron con el objetivo de proporcionar una nueva característica ó una nueva filosofía de diseño que otras distribuciones no proporcionaban y que otras han seguido posteriormente.

© 2004 Diego LÓPEZ ZAMARRÓN

3.2.- Clasificación

Mas que una clasificación, se proporciona información acerca del tipo de licencia bajo la que se distribuye (GPL, Comercial, ...), su estado actual (desarrollo, parada, ...) y su última versión del kernel en la que se basa, siempre con respecto a la fecha en que se desarrolló este documento, puesto que distribuciones que al día de hoy estén paradas (>2 años sin nuevas versiones) en un futuro pueden retomarse.

Distribución	***Licencia***	***Estado***	***Arquitectura***	***Última Versión kernel***
ADEOS	GNU/GPL	Activo	Nano-kernel	2.4.24 - --> 2.6.x
RTAI	GNU/GPL	Activo	Micro-kernel	2.4.xx - --> 2.6.x
RTLinux	GNU/GPL && Open RTLinux Patent	Parado	Micro-kernel	2.4.21
	Comercial	Desconocido	Micro-kernel	Desconocido
ART-Linux	Privado	Desconocido	Micro-kernel	Desconocido
KURT	Desconocida. Open Source	Activo	Extensiones POSIX	2.4.18
Linux/RK	Desconocida. Open Source	Activo	Recurso-kernel	2.4.18
Qlinux	GNU/GPL	Activo	QoS	2.4.4
RED-Linux	Desconocida	Desconocido	Extensiones POSIX	2.0.35
BlueCat-RT	Comercial. Open Source	Activo	Micro-kernel + Preemptable (kernel estándar)	Desconocido
RedHawk	Comercial	Desconocido	Preemptable-kernel	Desconocido
REDICE-Linux	Libre de Royaltis. Soporte comercial	Activo	Micro-kernel + QoS (kernel estándar)	2.4.18
TimeSys Linux	Comercial. Parte es GPL	Activo	Preemptable-kernel + Extensiones POSIX	2.4.21
Linux-SRT	GNU/GPL	Abandonada	Extensiones POSIX	2.2.15

Como se puede observar, en el mercado hay tanto distribuciones comerciales y de código abierto, como distribuciones libres. También se observa, que actualmente, las distribuciones libres ADEOS y RTAI son las que están más actualizadas porque están planeando sacar versiones validas para la versión 2.6 del kernel estándar de Linux. El resto ó están muy dejadas ó tardan bastante en sacar nuevas versiones. También se puede observar y se notará más adelante, que las distribuciones GNU/GPL proporcionan más información.

© 2004 Diego LÓPEZ ZAMARRÓN

3.3.- Descripción

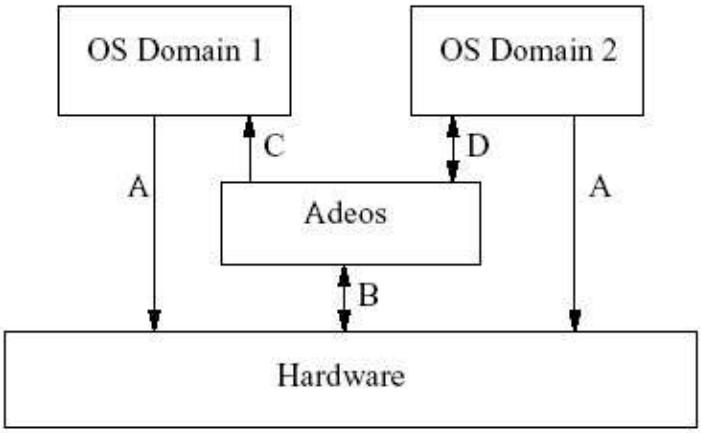
3.3.1.- ADEOS – Adaptative Domain Environment Operating Systems

- ¿Qué es ADEOS?

  El objetivo de ADEOS es proporcionar un entorno flexible para compartir los recursos hardware para múltiples sistemas operativos ó múltiples instancias de un mismo sistema operativo.

  ADEOS activa múltiples kernels, llamados dominios, que existen simultáneamente sobre el mismo hardware. Ninguno de éstos dominios necesariamente conoce la existencia del resto, pero todos ellos si conocen de la existencia de ADEOS. Un dominio puede ser un Sistema Operativo completo, pero no necesariamente.

  Actualmente ADEOS permite compartir las interrupciones hardware.



  - Licencia: GNU/GPL

- Fundamentos
  - Arquitectura

    La arquitectura de ADEOS es la de kernel dual y más específicamente la llamada Nano-kernel.

    - Dominios y Pipeline

      Para permitir que las interrupciones y los eventos del sistema sean repartidos para compartir por los múltiples kernels (normalmente SO completos), ADEOS define lo que ha llamado abstractamente “dominio”.

      Un dominio es un componente software del kernel base al cuál Adeos puede notificar:

      - Las interrupciones hardware.
      - Llamadas al sistema de las aplicaciones Linux.
      - Eventos del sistema lanzados por el kernel de Linux.
      - Otros eventos que personalicemos nosotros.

© 2004 Diego LÓPEZ ZAMARRÓN

Un dominio puede ser accesible como un módulo dinámico del kernel, ó como uno estático formando parte de una imagen del kernel, no produciendo ningún tipo de incidencia en el comportamiento de ADEOS.

ADEOS también puede asegurar que los eventos sean disparados en el orden correcto para los distintos dominios definidos, gracias a ello, es posible proporcionar determinismo.

Esto nos permite asignar a cada dominio una prioridad estática. El valor de esta prioridad determina el orden en que los eventos son tratados por los dominios. Todos los dominios son encolados de acuerdo a sus respectivas prioridades, formando un pipeline de forma abstracta, que es el que usa ADEOS para manejar el flujo de eventos desde el más prioritario hacia el menos prioritario. Los eventos de entrada son encauzados a la cabecera del pipeline (dominio más prioritario) y progresan hacia la cola (dominio menos prioritario).

El código del kernel de Linux es enteramente el sólo un dominio especial predefinido, que es creado por ADEOS en las primeras etapas de carga de Linux. Este dominio inicial normalmente hace referencia al “root” domain hasta que la infraestructura proporciona lo suficiente para cargar el resto de dominios dinámicamente (ej. el módulo cargador del kernel)

- Modo de Trabajo

ADEOS controla todas las interrupciones, los traps de la CPU y las excepciones del nivel hardware, ya que reprograma el software del manejador. De hecho, ADEOS actualmente reemplaza los manejadores instalados anteriormente por Linux durante el arranque por los suyos propios para las correspondientes interrupciones y eventos.

En la plataforma x86, se hace simplemente cambiando el descriptor de interrupción de la tabla.

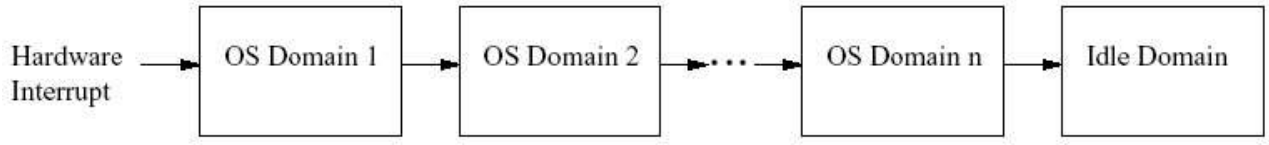
En cualquier momento dada una CPU con ADEOS activado, un dominio puede estar:

- Ejecutándose.
- Interrumpido por un dominio más prioritario durante el procesamiento de una interrupción/evento de entrada mientras el estaba procesando un evento pendiente.
- Voluntariamente autosuspendido, después de que todas las interrupciones/eventos hayan sido procesados.

Cuando un dominio ha finalizado de procesar una interrupción/evento que ha recibido, llama a un servicio especial de ADEOS (adeos_suspend_domain()) el cual provoca que se pase la interrupción/evento al siguiente dominio del pipeline y así sucesivamente, realizándose un ciclo del más prioritario al menos prioritario.

Hay que destacar que ADEOS reanuda un dominio sólo si tiene que procesar una interrupción/evento ó si ocurre que ha sido interrumpido por un dominio más prioritario que ha recibido alguna interrupción/evento para procesar, en otras palabras, no se produce intercambio entre dos dominios a menos que haya trabajo pendiente para alguno de ellos.

Cuando una interrupción ocurre en un sistema ocioso, ADEOS despierta al dominio más prioritario interesado en ella y el ciclo de proceso del pipeline comienza de nuevo.



© 2004 Diego LÓPEZ ZAMARRÓN

Ejemplo:

Dados los dominios A,B,C donde la prioridad es A>B>C, se puede decir que:

- Si A ejecuta, entonces B y C están interrumpidos ó suspendidos.
- Si B ejecuta, entonces A esta suspendido y C esta interrumpido ó suspendido.
- Si C ejecuta, entonces A y B deben estar suspendidos.

- ¿Por qué puedes necesitar ADEOS?
  - Porque necesitas controlar determinísticamente el flujo de interrupciones hardware usando una capa software antes de que el kernel de Linux las procese. Este control incluye la interceptación, el enmascarado y/o la priorización de las interrupciones.
  - Porque necesitas monitorizar las llamadas al sistema de Linux, añadiendo más información y sin tener que modificar el código de las llamadas al sistema.
  - Porque quieres un mecanismo para monitorizar los eventos internos que ocurren en el kernel de Linux como pueden ser la planificación de tareas, la creación de procesos ó las señales que capturan las tareas.

- Implementación de un dominio

  El interfaz entre un dominio y ADEOS esta compuesto de:

  - Un descriptor, que es una estructura de datos que describe las propiedades del dominio en el tiempo.
  - Una rutina de entrada al dominio, a la cual ADEOS llama para iniciar un dominio. Esta rutina se encarga de registrar el conjunto de interrupciones/eventos que va a manejar y a continuación el dominio pasa a ejecutar el bucle ocioso.

➢ Enlaces:

- ADEOS (web oficial distribución) --> http://www.opersys.com/adeos/
- ADEOS (web oficial desarrollo) --> https://gna.org/projects/adeos/
- Documentos:
  - Diseño Adeos: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/adeos.design.pdf
  - Adeos: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/porting.txt
  - RTOS sobre ADEOS: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtos.over.adeos.pdf

© 2004 Diego LÓPEZ ZAMARRÓN

3.3.2.- RTAI – Real Time Application Interface

- ¿Qué es RTAI?

  RTAI es una implementación de Linux para tiempo real basada en RTLinux. Añade un pequeño kernel de tiempo real bajo el kernel estándar de linux y trata al kernel de linux como una tarea con la menor prioridad. RTAI además proporciona una amplia selección de mecanismos de comunicación entre procesos y otros servicios de tiempo real.

  Adicionalmente, RTAI proporciona un módulo llamado LXRT para facilitar el desarrollo de aplicaciones de tiempo real en el espacio de usuario.

  - Licencia: GNU/GPL

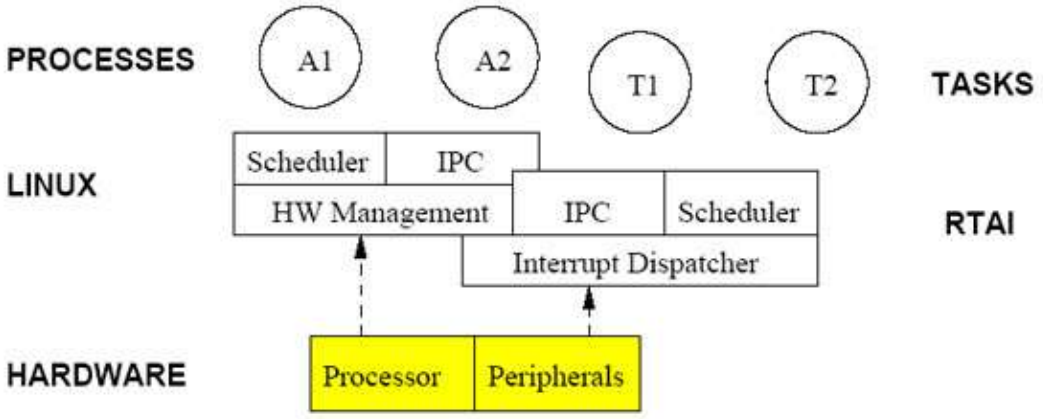
- Fundamentos

  - Arquitectura

    RTAI tiene una arquitectura similar a RTLinux. Al igual que RTLinux, RTAI trata el kernel estándar de Linux como una tarea de tiempo real con la menor prioridad, lo que hace posible que se ejecute cuando no haya ninguna tarea con mayor prioridad ejecutándose. Las operaciones básicas de las tareas de tiempo real son implementadas como módulos del kernel al igual que RTLinux. RTAI maneja las interrupciones de periféricos y son atendidas por el kernel de linux después de las posibles acciones de tiempo real que hayan podido ser lanzadas por efecto de la interrupción.

    La siguiente figura muestra la arquitectura básica de RTAI, que es muy similar a la de RTLinux. Las interrupciones se originan en el procesador y en los periféricos. Las originadas en el procesador (principalmente señales de error como división por cero), son manejadas por el kernel estándar, pero las interrupciones de los periféricos (como los relojes) son manejadas por RTAI Interrupt Dispatcher. RTAI envía las interrupciones a los manejadores del kernel estándar de linux cuando no hay tareas de tiempo real activas. Las instrucciones de activar/desactivar las interrupciones del kernel estándar son reemplazadas por macros que se enlazan con las instrucciones de RTAI. Cuando las interrupciones están desactivadas en el kernel estándar, RTAI encola las interrupciones para ser repartidas después de que el kernel estándar haya activado las interrupciones de nuevo.

    Adicionalmente, se puede ver en la figura el mecanismo de comunicación entre procesos (IPC), que esta implementado de forma separado por Linux y por RTAI. También existe un planificador (sheduler) distinto para Linux y para RTAI.



- HAL – Hardware Abstraction Layer

  Los desarrolladores de RTAI introducen el concepto de Real Time Hardware Abstraction Layer (RTHAL) que es usado para interceptar las interrupciones hardware y procesarlas después. RTHAL es una estructura instalada en el kernel de Linux que reúne los punteros a los datos internos del hardware relacionados en el kernel y las funciones necesarias por RTAI para operar. El objetivo de RTHAL es minimizar el número de cambios necesarios sobre el código del kernel y por tanto mejorar el mantenimiento de RTAI y del código del kernel de Linux. Con RTHAL, las diferentes operaciones (ej. manejar interrupciones) son más fáciles de cambiar ó modificar sin tener que interferir con la implementación de Linux. Por ejemplo, la estructura de RTHAL contiene la tabla de manejadores de interrupción, la cual es un lista de las funciones que son llamadas para manejar las diferentes interrupciones. El cambio consiste únicamente en modificar unas 20 líneas del código del kernel de Linux y añadir unas 50 nuevas líneas.

  Cuando RTHAL es instalado en Linux, las funciones y las estructuras de datos relacionada con la interacción con el hardware son reemplazada por punteros a la estructura de RTHAL.

- Planificación

  La unidades de planificación de RTAI son las tareas. Siempre hay al menos una tarea, llamada kernel de linux, que ejecuta como la tarea de menor prioridad. Cuando las tareas de tiempo real son añadidas, el planificador da entonces mayor prioridad a éstas sobre la tarea del kernel de Linux. El planificador proporciona servicios tales como suspend, resume, yield, make periodic, wait until, que son usadas en varios sistemas operativos de tiempo real.

  El planificador es implementado como un módulo del kernel dedicado (contrario a RTLinux) lo que facilita la implementación de planificadores alternativos si es necesario. Actualmente hay tres tipos de planificadores dependiendo del tipo de máquina:

  - Uniprocesador (UP)
  - Multiprocesador simétrico (SMP)

    Esta diseñado para máquinas SMP y proporciona un interfaz para las aplicaciones de forma que es posible seleccionar el procesador ó procesadores que deben ejecutar una tarea. Si el usuario no especifica un procesador para la tarea, SMP selecciona el procesador en función de la carga de trabajo.

  - Multi-Uniprocesador (MUP)

    Puede ser usado con ambos, pero al contrario que SMP, a las tareas se les debe especificar el procesador que deben usar. Viéndolo por el lado positivo, el planificador MUP permite unos mecanismo de tiempo más flexibles para las tareas que los planificadores SMP ó UP.

- Características de RTAI

  Con el objetivo de hacer el desarrollo de aplicaciones más fáciles y flexibles posibles, los desarrolladores de RTAI han introducido diferentes mecanismos para la comunicación entre procesos (IPC), entre las tareas de tiempo real y los procesos en el espacio de usuario. Como añadido al mecanismo IPC, RTAI proporciona servicios de manejo de memoria y threads compatibles con Posix.

  - Comunicación entre procesos (IPC – Inter-process comunication)

    RTAI proporciona una variedad de mecanismos para la comunicación entre procesos. Aunque los sistemas Unix proporcionan mecanismos similares a IPC para los procesos en el espacio de usuarios, RTAI necesita proporcionar una implementación propia para que las tareas de tiempo real puedan usar este mecanismo y no usen el estándar del kernel de Linux. Los diferentes mecanismo de IPC están incluidos como módulos de kernel, lo que facilita la carga cuando son necesarios. Como ventaja adicional el uso de módulos para los servicios IPC, facilita el mantenimiento y la expansión.

El antiguo mecanismo básico de comunicación de RTAI eran los FIFOs. FIFO es un asíncrono y no bloqueante canal de comunicación entre los procesos de Linux y las tareas de tiempo real. La implementación de RTAI de FIFO esta basado en la implementación de RTLinux, pero RTAI proporciona algunas características que no son posibles en RTLinux. Primeramente RTAI puede lanzar señales cuando hay eventos en el FIFO (escritura de nuevos datos). Los procesos en el espacio de usuario pueden entonces crear un manejador para la señal por los mecanismos estándar de Unix. Sin embargo, este mecanismo no es necesario para los procesos de usuario que quieran leer ó escribir del FIFO. Adicionalmente, puede haber múltiples lectores y escritores en el FIFO, cosa que no es posible en la versión de RTLinux. Finalmente, los identificadores FIFO pueden ser dinámicamente localizados con un nombre simbólico. Antes era necesario establecer un identificador global para el FIFO, lo que causaba problemas cuando múltiples e independientes procesos y tareas lo usaban.

Los semáforos es otra herramienta básica de sincronización entre procesos usada en los sistemas operativos. RTAI proporciona un API para usar semáforos, aunque cada semáforo esta técnicamente asociado a un FIFO, por tanto cada semáforo usa una entrada global del FIFO. Como añadido al servicio básico de semáforos, un semáforo puede estar asociado con un reloj, el cual puede ser usado para despertar un proceso encolado en un semáforo, incluso cuando el semáforo aun esta cerrado.

La compartición de memoria proporciona una alternativa a IPC y al paradigma FIFO, cuando un modelo de comunicación diferente es requerido. La memoria compartida es un bloque común de memoria que puede ser leída ó escrita por un proceso y una tarea en el sistema. Como los diferentes procesos pueden operar de forma asíncrona en la región de memoria, es necesario un diseño para asegurar que los datos no sean sobreescritos de forma intencionada.

Finalmente, el método más flexible de IPC quizás sean los mailboxes. Cualquier número de procesos pueden enviar y recibir mensaje de y desde un mailbox. Un mailbox almacena mensajes hasta un límite que se defina, y contrario a los FIFOs, mailbox preserva los mensajes que están en el límite. Puede haber un número arbitrario de mailbox activos en el sistema simultáneamente. RTAI también facilita la comunicación entre procesos mediante RPC.

- Gestión de memoria

  En las primeras versiones de RTAI la memoria tenía que ser asignada estáticamente y no era posible las asignación en tiempo real. Sin embargo en las actuales versiones se incluye un módulo gestor de memoria que permite la asignación dinámica de memoria por parte de las tareas de tiempo real usando un interfaz basado en una librería estándar de C.

  RTAI preasigna trozos de memoria antes de la ejecución de tiempo real. Cuando la tarea de tiempo real llaman a la función rt_malloc(), la respuesta que obtiene es el trozo preasignado. Antes de que el espacio se agote, RTAI reserva nuevos trozos de memoria (preasigna) para futuras llamadas. De manera similar ocurre con la función rt_free(), en este caso, se libera la memoria preasignada a la espera de futuras reservas. Cuando la suma de memoria liberada es mayor que un valor para una marca, se ordena su liberación.

- Threads Posix

  RTAI tiene módulos que proporcionan la implementación de threads de acuerdo al estándar POSIX 1003.1c. Usando las operaciones especificadas en el estándar, el usuario puede manejar de manera similar a como lo hace con los threads posix convencionales, excepto en cuanto a los conceptos de joining y detaching.

  Los threads de un mismo proceso comparten el espacio de memoria, por tanto es fácil el intercambio de información entre ellos, sin embargo, el uso de áreas de memoria compartida son necesarias para la sincronización. También se proporcionan los mecanismos de mutex y de variables de condición.

- LXRT: User-space Interface to RTAI

  LXRT es un API para RTAI que hace posible el desarrollo de aplicaciones de tiempo real en el espacio de usuario sin tener que crear módulos para el kernel. Esto es útil en primer lugar porque el espacio de memoria destinado al kernel no esta protegido de accesos inválidos, lo que puede provocar la corrupción de datos y el mal funcionamiento del kernel de Linux. En segundo lugar, si el kernel es actualizado, los módulos necesitan ser recompilados lo que puede provocar que sean incompatibles con la nueva versión.

  Mientras se desarrolla una aplicación de tiempo real en el espacio de usuario el programador puede usar las herramientas estándar de depuración hasta que ya no contienen errores, pero en este caso se trata de procesos de tiempo real flexibles (soft). Además, los servicios proporcionados por las llamadas al sistema de Linux están disponibles para las tareas. La ventaja de esto es que cuando la aplicación esté libre de errores puede convertirse a un módulo en el espacio del kernel, por lo que ya tendremos una tarea de tiempo real estricto. Sin embargo, al hacer esto, las llamadas al sistema utilizadas no sirven y deberán ser cambiadas por las proporcionadas por RTAI.

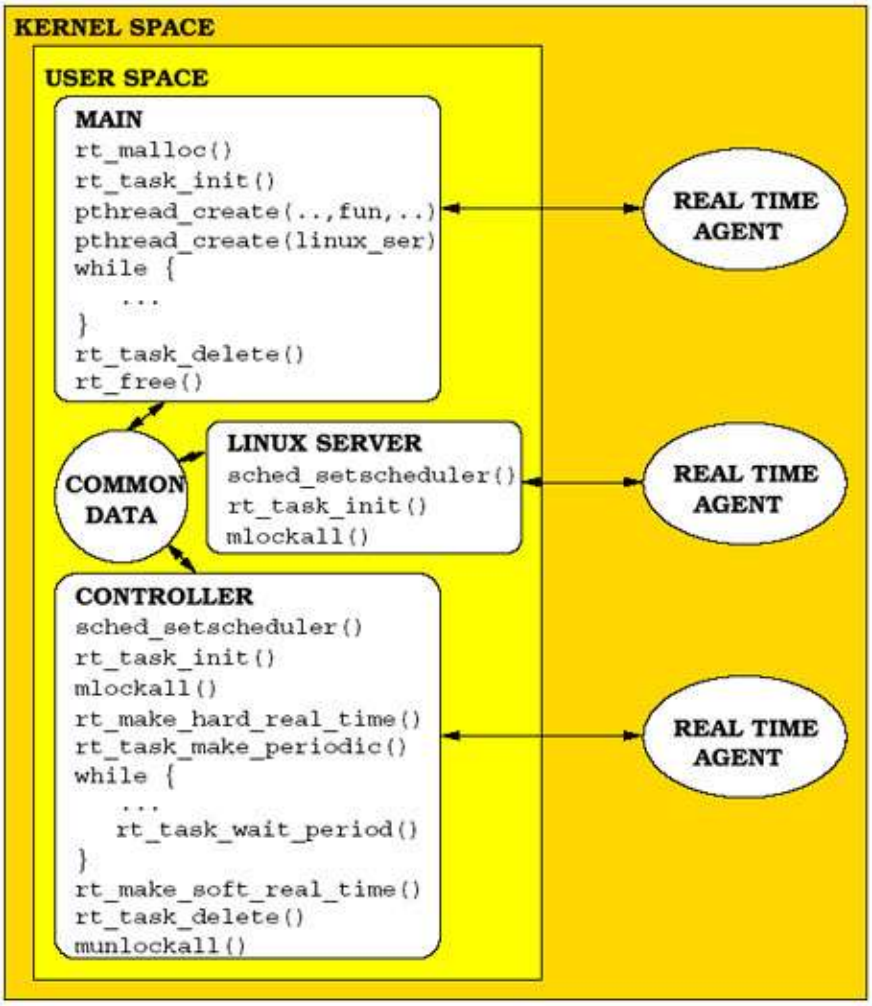
  El cambio de la aplicación de procesos del espacio de usuario a tareas de tiempo real es fácil porque LXRT proporciona un API simétrico para la comunicación entre procesos y otros servicios de RTAI, lo que significa que el mismo API puede ser usado por las tareas de tiempo real y por los procesos del espacio de usuario. El mismo API de LXRT puede ser también usado cuando 2 procesos del espacio de usuario ó 2 tareas de tiempo real se comunican entre si, esto implica que varios relojes y mensajes del sistema proporcionados por LXRT puedan ser usados incluso cuando la aplicación no requiera tiempo real.

  LXRT permite a las aplicaciones el intercambio dinámico entre tiempo real flexible y estricto mediante el uso de una simple llamada en el espacio de usuario. Cuando una aplicación esta en el modo de tiempo real flexible, usa el planificador de Linux, pero requiere el uso de la política de planificación FIFO. Sin embargo la planificación de procesos FIFO puede provocar la perdida de ranuras de tiempo por varias razones, por ejemplo, porque se esta ejecutando un manejador de interrupciones o el planificador de tareas de RTAI.

  Con el objetivo de facilitar el paso a tiempo real estricto, LXRT crea un agente en el espacio del kernel para cada proceso del espacio de usuario con requerimientos de tiempo real. Cuando el proceso entra en modo de tiempo real estricto, el agente desactiva las interrupciones, y mueve al proceso fuera del planificador de Linux y lo añade a la cola del planificador de RTAI. Ahora el proceso no es interrumpido por las interrupciones de otro proceso Linux. Para poder realizar este cambio el proceso debe asegurar que la memoria usada por el proceso este en la memoria RAM y debe desactivar la paginación usando la función mlockall(). Esta llamada no debe ser usada en modo tiempo real estricto.

  Aunque los desarrolladores de RTAI ven bien el desarrollo de aplicaciones de tiempo real estricto usando LXRT, el tiempo de respuesta no es tan bueno como la ejecución de las tareas como módulos del kernel.

© 2004 Diego LÓPEZ ZAMARRÓN



- Enlaces:
  - RTAI (web oficial) --> http://www.aero.polimi.it/~rtai/index.html
  - RTAI (artículo) --> http://www.linuxdevices.com/articles/AT6605918741.html
  - Documentos:
    - Historia: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/history.pdf
    - RTAI sobe ADEOS: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtai.over.adeos.pdf
    - Diseño RTAI: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtai.pdf
    - Otro: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtdistributedcontrolsystem.rtai.ppt

© 2004 Diego LÓPEZ ZAMARRÓN

3.3.3.- RTLinux

- ¿Qué es RTLinux?

  RTLinux es un sistema operativo de tiempo real que ejecuta Linux como un thread de menos prioridad que las tareas de tiempo real. Con este diseño, las tareas de tiempo real y los manejadores de interrupciones nunca se ven retrasados por operaciones que no son de tiempo real.

  La primera versión de RTLinux estaba diseñada para ejecutarse en la plataforma x86 y proporcionaba una pequeña API y un pequeño entorno de programación. La versión 2, que fue totalmente reescrita, fue diseñada para el soporte de multiprocesamiento simétrico (SMP) y para ser ejecutada en una amplia variedad de arquitecturas.

  RTLinux proporciona la capacidad de ejecutar tareas de tiempo real y manejadores de interrupciones en la misma máquina que el Linux estándar. Estas tareas y los manejadores ejecutan cuando se necesitan en detrimento de lo que estuviera ejecutando Linux. El peor caso de tiempo es entre que se detecta la interrupción hardware y el procesador ejecuta la primera instrucción del manejador de la interrupción. Este tiempo es del orden de los 10 microsegundos en la plataforma x86.

  - Licencia: Actualmente hay 2 versiones de RTLinux:
    - RTLinux/Open: Disponible bajo la licencia GPL, pero que ya no se trabaja en ella desde 2001.
    - RTLinux/Pro: Distribución comercial.

    Ambas versiones son distribuidas por la empresa FSMLabs.

    La arquitectura de diseño de RTLinux se encuentra patentada por FSMLabs. Una versión de la patente se encuentra accesible aquí: http://www.fsmlabs.com/products/rtlinuxpro/rtlinux_patent.html

    Y aquí se puede ver que la FSF (Free Software Foundation) esta de acuerdo en los términos de la patente: http://www.gnu.org/philosophy/rtlinux-patent.html

    A través de este otro enlace podemos leer una pequeña reflexión sobre los efectos de la patente en el desarrollo de aplicaciones en RTLinux: http://www.linuxdevices.com/articles/AT2094189920.html

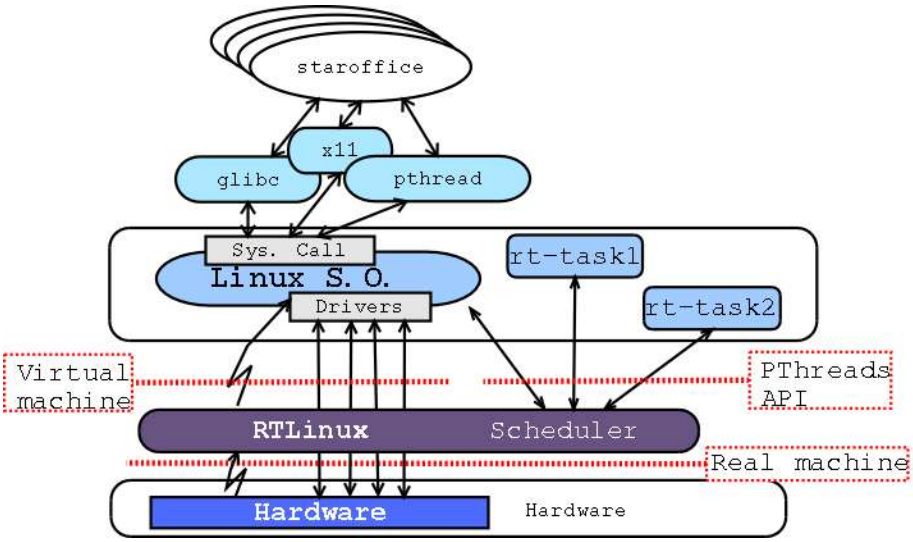
- Fundamentos

  A partir de aquí la descripción que se hace es sobre RTLinux/Open o RTLinux/GPL.

  - Arquitectura

    RTLinux es un pequeño y rápido sistema operativo que sigue el estándar POSIX 1003.13: sistema operativo de tiempo real mínimo.

    RTLinux añade una capa de abstracción hardware entre el kernel estándar de Linux y el hardware de la máquina. Esta es la arquitectura de micro-kernel.



Hay 3 modificaciones principales en el kernel de Linux con el objetivo de que RTLinux tenga el control del hardware de la máquina:

- Control directo de las interrupciones hardware.
- Control del reloj hardware e implementación de un reloj virtual para Linux.
- El control de las interrupciones por parte de Linux es reemplazado por 2 funciones que permiten activar ó desactivar las interrupciones, pero las virtuales.

Estas modificaciones del kernel de Linux son complejas aunque no requieren excesivo código, pero si más que RTAI.

RTLinux proporciona un entorno de ejecución bajo el kernel de Linux, como consecuencia de esto, las tareas de tiempo real no pueden usar los servicios de Linux. Para disminuir este problema, el sistema de tiempo real se ha divido en dos partes: la capa de tiempo real estricto que ejecuta encima de RTLinux, y la capa de tiempo real flexible, que ejecuta como un proceso normal de Linux. Para la comunicación de ambas capas se pueden usar varios mecanismos (FIFO, memoria compartida).

La propuesta de 2 capas es un método útil para proporcionar tiempo real estricto mientras se mantienen las características de escritorio de un sistema operativo. La separación del mecanismo del kernel de tiempo real del mecanismo del kernel de Linux de propósito general permite optimizar ambos sistemas de forma independiente.

© 2004 Diego LÓPEZ ZAMARRÓN

- Características:
  - Soporte de múltiples arquitecturas y válida para arquitecturas multiprocesador.
  - Gestión de procesos: Planificación, soporte de threads periódicos, amplio rango de prioridades, creación y borrado de threads, etc.
  - Gestión de memoria: No protección de memoria en el kernel y no asignación de memoria de forma dinámica.
  - Comunicación entre procesos: Semáforos, Mutex, control de inversión de prioridades, memoria compartida y FIFO's.
  - Tiempo y relojes: Resolución de nanosegundos. No relojes de usuario. Facilidades para añadir nuevos relojes hardware.
  - Programación de drivers: Se proporcionan funciones de acceso a dispositivos.
  - No proporciona herramientas de calidad de servicio.

➢ Enlaces

- Página oficial:
  - http://www.fsmlabs.com
  - http://www.rtlinux.com
  - http://www.rtlinux.org
- Repositorio RTLinux GPL: http://www.rtlinux-gpl.org
- Análisis RTLinux/GPL: http://www.mnis.fr/opensource/ocera/rtos/c1450.html
- Documentos:
  - Introducción RTLinux : http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtlinux_intro.pdf
  - Diseño RTLinux v2: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/design_rtl_v2.pdf
  - RTLinux/Pro:
    - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtlpro.pdf
    - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtlpro.datasheet.pdf
  - Instalación (Manual – español)
    - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/man_instal_RTLinux.pdf
    - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/man_instal_RTLinux.htm
  - Instalación (Guías rápidas – ingles)
    - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/kickstart.2.4.18.pdf
    - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/kickstart.2.4.21.pdf
  - Control de procesos: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtlinux.and.process.control.security.pdf

© 2004 Diego LÓPEZ ZAMARRÓN

3.3.4.- ART Linux

- ¿Qué es ART Linux?

  ART Linux es una extensión de tiempo real sobre Linux basado en RTLinux y desarrollado por Youichi Ishiwata.

  - Licencia: ART Linux es un parche sobre el kernel estándar de Linux pero que actualmente no se distribuye bajo la licencia GPL ya que contiene código desarrollado por Ishiwata en exclusiva. El que no se distribuya bajo la licencia GPL es debido a que Ishiwata trabaja para el gobierno y existe una ley de propiedad intelectual del gobierno.

- Fundamentos

  - Arquitectura

    La arquitectura es similar a la de RTLinux por tanto tienen arquitectura de kernel dual.

  - Ventajas de ART Linux

    - ART Linux permite reusar los drivers de dispositivos existentes en el Linux estándar así como las aplicaciones.
    - Compatibilidad a nivel de fuente de los drivers con el Linux estándar. Es decir si recompilas el driver, puede seguir siendo usado por las tareas de tiempo real.
    - No inversión de prioridades. Se resuelven automáticamente los problemas de inversión de prioridades.
    - Compatibilidad a nivel binario de los programas de usuario con el Linux estándar. No es necesario recompilar los programas con el objetivo de que sean usados por las tareas de tiempo real.
    - Evita indeterminismo en la ejecución de tiempo real debido a las interrupciones ya que trata las interrupciones con una estrategia periódica.
    - Las tareas de tiempo real ejecutan en modo usuario pero privilegiado, de forma que se beneficia de la protección de memoria y por tanto las tareas de tiempo real son seguras y no pueden provocar fallos en el kernel.

- Enlaces

  - http://www.linuxdevices.com/links/LK6839947367.html

3.3.5.- KURT – Kansas University Real Time

- ¿Qué es KURT?

  KURT junto con UTIME es una extensión al kernel estándar de Linux para proporcionar, bajo demanda, resolución de microsegundos en los relojes y planificación de tiempo real.

  - Licencia: Desconocida. Open source.

- Fundamentos
  - Arquitectura de KURT
    - UTIME

      El kernel estándar de linux ofrece 10ms de resolución en la sincronización. Para la mayoría de las aplicaciones este nivel de granularidad es suficiente, pero no para las aplicaciones de tiempo real. UTIME es una modificación al kernel de Linux para proporcionar una resolución entorno a los 10 microsegundos.

      Para llevarlo a cabo usa varios métodos:

      - Llamada al sistema “nanosleep”. Mediante esta llamada se permite que los procesos duerman con una resolución de microsegundos.
      - Temporizadores. Cada proceso tiene acceso a 3 temporizadores. Cuando vencen se envía una señal al proceso. Gracia a UTIME es posible configurar los intervalos de tiempo del temporizador con resolución de microsegundos.
      - Llamada al sistema “select”. Esta llamada espera a que varios descriptores de ficheros cambien de estado y en el kernel estándar esta limitada esa espera a un resolución de 10 ms. Gracias a UTIME, permite configurar con resolución de microsegundos.
      - Llamada al sistema “poll”. Similar a la anterior, permite configurar con resolución de microsegundos.
      - Módulo por defecto del kernel. Linux permite que nosotros mismo definamos nuestros propios módulos para el kernel, con lo que podemos añadir nuestros propios relojes y sus manejadores.

© 2004 Diego LÓPEZ ZAMARRÓN

- KURT

  KURT proporciona un manejador de eventos con la facilidad de un planificador de tiempo real con resolución de microsegundos gracias a UTIME.

  El kernel estándar de Linux proporciona 3 políticas de planificación diferentes, que son: SHED_FIFO, SHED_RR y SHED_OTHER, que son suficientes para la mayoría de aplicaciones que no requieren una granularidad de resolución muy fina. Un valor alto de prioridad no garantiza a un proceso su planificación, por eso KURT proporciona un mecanismo de planificación bajo demanda para garantizar a los proceso el acceso a los recursos que necesite y en un orden explicito.

  Todas las interacciones con el subsistema KURT se realizan a través de un psudodispositivo (/dev/kurt), que únicamente proporciona 3 operaciones: open, close y ioctl. Cada vez que se quiera usar KURT simplemente hay que abrir una instancia del dispositivo y cuando se termina, simplemente se cierra.

  Para facilitar la programación de aplicaciones KURT dispone de un API que proporciona las siguientes 3 categorías de rutinas:

  - Operaciones generales y de utilidad. Un ejemplo de utilidad es la función “kurt_open()”, que abre una instancia del psudodispositivo.
  - Inicialización, registro y control de procesos. Un ejemplo es la función “rt_suspend()”, que suspende un proceso por un determinado plazo de tiempo. Antes de registrar los procesos de tiempo real se debe determinar que tipo de planificación van a soportar, puesto que KURT soporta un amplio abanico de modos y niveles de planificación y combinación entre ellos.
  - Planificación. Un ejemplo es la función “set_sheduling_task()”, que registra los procesos actuales en el planificador de tareas de KURT.

  Para usar este API en los programas de usuario simplemente hay que incluir el fichero de cabecera linux/rt.h del kernel de KURT en el archivo. Más información en el manual de usuario.

- UTIME/KURT frente a RTLinux

  KURT es capaz de planificar cada tarea en un nivel distinto del planificador según que requerimientos de tiempo real necesite, frente a RTLinux / RTAI que planifica todas las tareas de tiempo real en un planificador ejecutivo.

➢ Enlaces

- KURT (web oficial) --> http://www.ittc.ku.edu/kurt/
- Documentos:
  - Manual de usuario: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/user.manual.pdf

3.3.6.- Linux/RK – Linux/Resource Kernel

- ¿Qué es Linux/RK?

  Linux/RK significa Linux / Resource Kernel y consiste en incorporar extensiones de tiempo real a Linux mediante una abstracción llamada recurso kernel.

  - Licencia: Desconocida. Open source

- Fundamentos
  - Arquitectura

    Las aplicaciones que requieran acceder a alguno de los recursos se comunica con el recurso kernel y realiza una reserva. Si es aceptada obtendrá el recurso cuando lo haya solicitado.

    Un recurso kernel es un kernel de tiempo real que proporciona el oportuno y garantizado acceso a los recursos del sistema para las aplicaciones que lo requieran.

  - Características

    Actualmente las actividades que soporta Linux/RK son:

    - Reserva de ancho de banda en acceso a los discos.
    - Reserva de ancho de banda de la red.
    - Co-planificación de varios recursos.
    - Integración con Java para tiempo real.
    - Lista de control de recursos.

- Enlaces
  - Página oficial: http://www-2.cs.cmu.edu/afs/cs/project/art-6/www/
  - Página del Coordinador: http://www-2.cs.cmu.edu/~rajkumar/linux-rk.html
  - Documentos:
    - Resource Kernel: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/rtss.ps
    - Resouce Kernel Multimedia: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/mmcn.ps

3.3.7.- Qlinux

- ¿Qué es Qlinux?

  Qlinux es un kernel de Linux que proporciona calidad de servicio garantizada para requerimientos de tiempo real flexible. Por tanto no es un sistema operativo específico de tiempo real, sino que esta orientado hacia aplicaciones multimedia que requieren una determinada calidad de servicio.

  - Licencia: GNU/GPL

- Fundamentos

  - Arquitectura

    Usa una arquitectura precursora a la de "kernel preemptable", pero en este caso no solo aplicada al acceso de la CPU sino que también aplicada al acceso a la red y al disco.

  - Características

    La última versión de Qlinux se basa en el kernel 2.4.4 e incluye las siguientes características:

    - H-SFQ(Hierarchical Start Time Fair Queuing) para el planificador de la CPU. El planificador activa una planificación jerárquica de forma que asigna ancho de banda de la CPU de forma justa entre las aplicaciones ó clases de aplicaciones.
    - H-SFQ para el planificador de paquetes de red. De manera similar a antes, pero en este caso, proporciona transferencias garantizadas y una asignación de ancho de banda para los paquetes de flujos individuales ó clases de flujos.
    - Algoritmo de planificación del disco (Cell disk sheduler). Soporta múltiples clases de aplicaciones tales como interactivas, best-effort, transferencias intensivas, tiempo real flexible, etc. [No estable]

    Cuando se activa Qlinux estas características reemplazan a las disponibles en el kernel estándar de Linux. Las actuales versiones proporcionan flexibilidad permitiendo la combinación de estas características según se necesiten.

- Enlaces

  - Documentos:
    - Soporte Multimedia: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/os.support.multimedia.ppt
    - Aplicaciones Multimedia en Qlinux: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/application.qlinux.ppt
  - Página oficial: http://lass.cs.umass.edu/software/qlinux/
  - Artículo: http://www.linuxdevices.com/links/LK8255146646.html

3.3.8.- RED-Linux

- ¿Qué es RED-Linux?

  RED-Linux es una versión de Linux para tiempo real y embebida. Proporciona capacidades adicionales al kernel estándar de Linux para proporcionar comportamiento de tiempo real.

  - Licencia: Desconocida

- Fundamentos

  RED-Linux soporta:

  - Un núcleo pequeño.
  - Un tiempo de respuesta rápido para las tareas.
  - Modularidad y Planificador de CPU reemplazable en tiempo de ejecución.
  - Un marco general de planificación (GSF-General Sheduling Framework).

  En teoría se puede usar bajo cualquier distribución de Linux, pero recomiendan usar Debian.

- Enlaces
  - RED-Linux (Web oficial) [No funciona]: http://linux.ece.uci.edu/RED-Linux/index.html
  - Artículo: http://www.linuxdevices.com/links/LK7432493143.htm
  - Tolerancia a fallos usando RED-Linux:

    http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/tolerancia.fallos.redlinux.pdf

3.3.9.- BlueCat RT

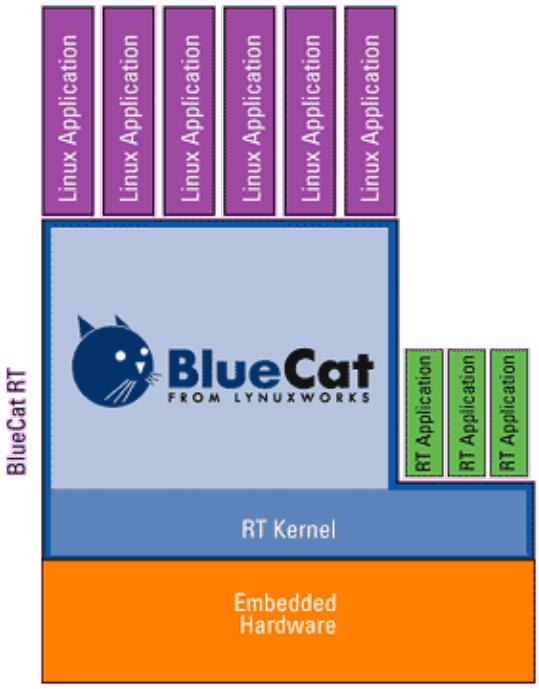
- ¿Qué es BlueCat RT?

  BlueCat RT es una versión de Linux para tiempo real y embebida. Es una solución híbrida entre el sistema operativo embebido BlueCat (LinuxWorks) y el sistema RTLinux/Pro (FSMLabs).

  - Licencia: Comercial. Open source

- Fundamentos

  - Arquitectura



  - Ventajas de BlueCat RT
    - Asegura el manejo de interrupciones con tiempo crítico y otras operaciones hardware de bajo nivel para implementar el kernel de Linux como un thread pequeño y de alta respuesta en un sistema operativo de tiempo real. Este mini sistema operativo de tiempo real captura las interrupciones y otras funcionalidades de bajo nivel para el procesamiento preliminar antes de pasárselas al thread que ejecuta el kernel de Linux.
    - Los dos sistemas operativos forman una solución híbrida que ejecuta en un solo procesador, donde la mayoría de las aplicaciones operan bajo el kernel estándar de Linux y un subconjunto hace uso de las características de tiempo real de RTLinux Pro, que forma el kernel de tiempo real. Esto hace de BlueCat una solución ideal para aquellas aplicaciones en las que sus tareas no requieran tiempo real la mayor parte del tiempo.

- Incluye un API de tiempo real compatible con POSIX y con IEEE 1003.13, lo que permite que aplicaciones que no requieren una completa funcionalidad de Linux puedan ser ejecutadas en esta capa.
- Basado en estándares abiertos y en código abierto, lo que facilita la migración de aplicaciones hacia BlueCat RT.

- Características
  - Estabilidad

    Es una distribución estable porque se basa en un sistema operativo embebido con cierto grado de comercialidad y por tanto se diseña para que sea flexible, escalable y con productividad inmediata.
  - Flexibilidad

    Soporte de un completo rango de configuraciones, desde pequeños dispositivos para el consumidor hasta grandes sistemas multiprocesadores.

➢ Enlaces
- BlueCat RT: http://www.lynuxworks.com/products/bluecat-rt/bluecat-rt.php3
- BlueCat & LinuxWokrks (web oficial):
  - http://www.bluecat.com/
  - http://www.lynuxworks.com/
- Artículo: http://linuxdevices.com/products/PD7095580223.html
- Artículo: http://www.electronicstalk.com/news/lyn/lyn124.html
- Documentos:
  - linxOS: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/linxos.pdf
  - BlueCatRT: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/bluecat-rt.pdf
  - Ambos: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/linuxworks.pdf

© 2004 Diego LÓPEZ ZAMARRÓN

3.3.10.- RedHawk

- ¿Qué es RedHawk?

  Es una versión de tiempo real del código abierto de Linux desarrollada por la empresa Concurrent Computer Corporation's RedHawk bajo los estándares industriales y POSIX. RedHawk es compatible con la distribución Red Hat Linux proporcionando alto rendimiento de entrada/salida, tiempo de respuesta garantizado a evento externos y comunicación entre procesos optimizada.

  RedHawk es ideal para ambientes con complejas aplicaciones de tiempo real como pueden ser la simulación, la adquisición de datos ó los sistemas de control industrial.

  - Licencia: Comercial

- Fundamentos

  - Arquitectura

    El sistema de administración, las utilidades y los comandos del nivel de usuario son los estándar proporcionados por Red Hat, en cambio, para lograr rendimiento de tiempo real, reemplaza el kernel de Red Hat por uno que es multithread, con un kernel completamente apropiable, es decir capaz de interrumpir a tareas con menor prioridad, y de baja latencia.

    RedKawk proporciona soporte al multiprocesamiento simétrico, que incluye un control de carga y protección de CPU para maximizar el determinismo y el rendimiento de tiempo real en soluciones críticas.

  - Características

    - Ambiente completo de desarrollo. Ofrece un conjunto completo de herramientas de desarrollo, desde compiladores a debuggers y herramientas que muestran la traza de ejecución.
    - SMP escalable y protección de procesador.
    - Multithread y kernel prioritario.
    - Planificador basado en la frecuencia.
    - Reloj de tiempo real y módulo de interrupciones.
    - Herramientas de desarrollo de aplicaciones de tiempo real.
    - Depurador de código fuente.
    - Analizador de eventos mediante traza de código.
    - Simulador de un planificador.
    - Monitor de datos de prueba.

- Enlaces

  - RedHawk (web oficial): http://www.ccur.com/isd_solutions_redhawklinux.asp?o9=1
  - Artículo: http://www.linuxdevices.com/products/PD3887955999.html
  - Documentos:
    - RedHawk: http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/redhawk.linux.pdf

3.3.11.- REDICE-Linux

- ¿Qué es REDICE-Linux?

  REDICE-Linux es un kernel de tiempo real que combina la planificación de tiempo real estricto y un kernel prioritario con la baja latencia de un completo kernel de tiempo real estricto como es el de RTAI. Es decir, combina ambas tecnologías de diseño de sistemas operativos de tiempo real.

  REDICE-Linux es un sistema operativo embebido ideal para aplicaciones de internet, dispositivos de redes, dispositivos médicos, industriales, adquisición de datos, control de máquinas y otros donde el tiempo es crítico, además de aplicaciones que requieran una determinada y garantizada calidad de servicio.

  - Licencia: Libre de Royalties. Open Source. Soporte es de pago.

- Fundamentos

  - Arquitectura

    Esta diseñada con una arquitectura dual, por una parte utiliza la arquitectura de microkernel, es decir tiene un segundo kernel, que es el de RTAI, y por la otra, modifica el kernel estándar de Linux para que se convierta en "kernel preemptable".

  - Ventajas
    - Ahorro de dinero: Usando los recursos y el sistema de control de presupuesto de los componentes de REDICE, los usuarios pueden construir distintas funcionalidades a bajo coste.
    - Fiabilidad: Todas las tareas tienen su propio control de los recursos y son independientes las unas de las otras, como resultado de esto, el comportamiento inadecuado de algunas de las tareas no afecta a la estabilidad del sistema.
    - Pronosticable: REDICE-Linux proporciona un comportamiento más predecible mediante el uso de una guía de tiempos integrada y un paradigma de planificación compartido.
    - Sensibilidad: Las tareas no pueden ser bloqueadas por otros trabajos menos críticos por un periodo de tiempo indefinido.

© 2004 Diego LÓPEZ ZAMARRÓN

- Características
  - REDICE-Linux proporciona un completo software de desarrollo, y de herramientas para el rápido desarrollo de sistemas de tiempo real embebidos.
  - Reloj con alta precisión del orden de microsegundos.
  - Sistema de planificación potente y predecible.
  - Mecanismo para garantizar rendimiento a las tareas.
  - Potencia y fiabilidad del código abierto de Linux.
  - Sistema de tiempo real con capacidad de garantizar a las aplicaciones rendimiento y ejecución de tareas con latencias del rango de los microsegundos.

➢ Enlaces

- REDSonic (Web oficial): http://www.redsonic.com/products/redice.htm
- Artículos:
  - http://www.linuxdevices.com/products/PD7480524149.html
  - http://www.dedicated-systems.com/encyc/buyersguide/rtos/Object261.html

3.3.12.- TimeSys Linux

- ¿Qué es TimeSys Linux?

  TimeSys Linux es más que una distribución de un kernel de tiempo real, es un conjunto completo de herramientas para el desarrollo de aplicaciones de tiempo real de forma cómoda y rápida. Es una distribución para sistemas embebidos.

  - Licencia: Comercial, menos algunos elementos que se distribuyen bajo la licencia GNU/GPL.

- Fundamentos
  - Arquitectura

    TimeSys Linux se basa en la arquitectura de "kernel preemptable" para proporcionar rendimiento predecible y latencia limitada.

  - Características
    - En el peor de los casos, tiempo de respuesta de 100 microsegundos.
    - Soporte en el mismo sistema, de tiempo real estricto y tiempo real flexible.
    - Soporte para aplicaciones Linux estándar y con interfaces POSIX, así como relojes con resolución de microsegundos (dependiendo del hardware).

  - Ventajas:
    - Flexibilidad, fiabilidad y fuerza del código abierto de Linux.
    - Mejor control sobre el rendimiento de las aplicaciones que otras distribuciones de Linux.
    - Mejora de la predicción para las aplicaciones embebidas actuales.
    - Soporte empotrado para el desarrollo de aplicaciones tanto en ordenadores Linux como Windows.
    - Escalabilidad para futuros requerimientos de rendimiento.

- Enlaces
  - TimeSys (Página principal): http://www.timesys.com/
  - TimeSys Notas kernel: http://www.timesys.com/index.cfm?bdy=linux_bdy.cfm
  - TimeSys Kits desarrollo: http://www.timesys.com/index.cfm?bdy=linux_bdy_home.cfm
  - Artículo: http://www.linuxdevices.com/articles/AT5625209055.html

© 2004 Diego LÓPEZ ZAMARRÓN

3.3.13.- Linux-SRT

- ¿Qué es Linux-SRT?

  Linux-SRT es una extensión al kernel de Linux que inició el camino hacia la ejecución de aplicaciones de tiempo real. El kernel estándar de Linux no era un sistema operativo multimedia, no garantizaba que el audio, el vídeo u otro proceso crítico ejecutará con una transferencia fijada, por eso nació este proyecto con el objetivo de crear un sistema operativo multimedia que proporcionara una calidad de servicio determinada, pero no estricta, de ahí el nombre, Linux-SRT (soft real time). No es adecuado para sistemas críticos. Este proyecto se ha abandonado actualmente debido al laborioso proceso de modificación del kernel.

  - Licencia: GNU/GPL

- Fundamentos

  - Arquitectura

    La arquitectura de Linux-SRT es similar a la de KURT, es decir, añadir al kernel estándar requerimientos de tiempo real mediante la inclusión de librerías conforme al estándar POSIX 1003.13.

  - Características
    - Planificación: El paquete correspondiente al planificador permite especificar términos de calidad de servicio, como por ejemplo el uso de un determinado porcentaje máximo a cada tarea de tiempo real. Esto es útil por ejemplo para grabar cds ó escuchar mp3's sin interrupción.
    - Gráficos: La asignación de CPU a una tarea no es el único factor que afecta a la velocidad de las aplicaciones, también lo es el servidor X, por eso se modifica para priorizar la renderización de gráficos en función de los parámetros de planificación de cada cliente X.
    - Interfaz de usuario: Permite la configuración de parámetros de planificación a nivel de usuario.

- Enlaces
  - Página proyecto: http://www.srcf.ucam.org/~dmi1000/linux-srt/
  - Artículo: http://www.linuxdevices.com/links/LK9065547342.html

© 2004 Diego LÓPEZ ZAMARRÓN

4.- Elección de distribución/es

Para nuestro proyecto hemos decidido usar las distribuciones ADEOS M3 y RTAI 24.1.12 sobre una distribución Debian Linux. El kernel estándar sobre el que trabajaremos es la versión 2.4.21.

Hemos decidido elegir estas distribuciones por varias razones:

- Ambas, así como Linux, se distribuyen bajo la licencia GNU/GPL.
- Ambas distribuciones, al día de hoy, se actualizan con rapidez.
- ADEOS nos permite tener un control de todos los eventos que ocurren en el sistema, tanto hardware como Software.
- RTAI nos proporciona un entorno de desarrollo flexible que nos permite desarrollar inicialmente las aplicaciones en el espacio de usuario, sin riesgo para el kernel, y posteriormente crear el módulo del kernel libre de errores.

4.1.- Manuales de Instalación

4.1.1.- Instalación de ADEOS

- Obtención de la distribución

  En primer lugar debemos comprobar que versión del kernel de Linux tenemos instalada en nuestro ordenador. Se hace a través del siguiente comando:

  ```
  root@hostname:~$ uname -r
  2.4.20
  ```

  En nuestro caso tenemos la versión 2.4.20, pero la instalación la vamos a hacer sobre la versión 2.4.21, por lo que tenemos que obtener el código fuente. Se puede obtener a través de http://www.kernel.org. Una vez localizado lo guardamos en `/usr/src`.

  En el caso de querer seguir usando la versión actual que tenemos del kernel debemos comprobar que tenemos el código fuente, puesto que va a ser necesario parchear el kernel, para ello accedemos al directorio `/usr/src`. Si tenemos el código fuente existirá un directorio con el nombre `linux-2.x.x` (el correspondiente a la versión que tengamos) y un directorio con el nombre `linux` que es un enlace simbólico al anterior directorio. Si no existen deberemos obtener el código fuente desde el enlace anterior.

  La versión ADEOS-M3 la podemos obtener desde http://download.gna.org/adeos/milestones/adeos-m3.tar.gz. ADEOS-M3 es la última versión estable que contiene parches para distintas versiones, además de diversa documentación. Si no queremos bajarnos toda la distribución y únicamente queremos el parche para nuestra versión del kernel lo podemos localizar en http://download.gna.org/adeos/patches/ . Una vez localizado lo guardamos en `/usr/src`.

- Instalación

  - Descomprimimos el código fuente del kernel de Linux

    ```
    root@hostname:~$ cd /usr/src
    ```

    Si esta en formato tar.gz:

    ```
    root@hostname:/usr/src$ tar -xzf linux-2.4.21.tar.gz
    ```

Si esta comprimido con bzip2:

```
root@hostname:/usr/src$ tar -xjf linux-2.4.21.tar.bz2
```

Al realizar esto se ha creado un directorio llamado linux-2.4.21. Lo renombramos para identificarlo como kernel parcheado con ADEOS.

```
root@hostname:/usr/src$ mv linux-2.4.21 linux-2.4.21-adeos-m3
```

Ahora creamos un enlace simbólico a este directorio:

```
root@hostname:/usr/src$ ln -s linux-2.4.21-adeos-m3/ linux
```

Se ha creado un directorio llamado `linux` que es un enlace simbólico al directorio `linux-2.4.21-adeos-m3.`

- Descomprimimos la distribución ADEOS-M3

```
root@hostname:/usr/src/$ tar -xzf adeos-m3.tar.gz
```

Se ha creado un directorio llamado `adeos-m3`.

- Parcheado del kernel

Primero comprobamos que el parche que vamos a aplicar es valido. Para ello ejecutamos:

```
root@hostname:/usr/src$ cd linux

root@hostname:/usr/src/linux$ patch -p1 –dry-run < ../adeos-m3/patches/linux/adeos-linux-2.4.21-r8.patch
```

Si no da ningún error aplicamos el parche:

```
root@hostname:/usr/src/linux$ patch -p1 < ../adeos-m3/patches/linux/adeos-linux-2.4.21-r8.patch
```

La localización del parche puede ser distinta si únicamente nos hemos descargado el parche.

- Ahora debemos configurar y compilar el nuevo kernel parcheado

Primero configuramos el kernel. A parte de la configuración correcta del hardware, en especial el tipo de procesador, es necesario que en el apartado 'General Setup' configuremos:

```
root@hostname:/usr/src/linux$ make menuconfig

General Setup -->

   [*] Networking support
   <M> Adeos support (NEW)
   [*] Adeos domain mutex support (NEW)
   [*] PCI support
```

Una vez terminado se guarda la configuración en un fichero llamado `.config`. Es recomendable guardar la configuración en otro fichero (`cp .config config_adeos`) como seguridad si tenemos que aplicar el comando `mrproper`. Este comando se debe usar sólo si ya teníamos el código fuente y le habíamos configurado varias veces ó habíamos parcheado el kernel antes. Nosotros al partir de un código fuente limpio no es necesario.

Ahora compilamos y obtendremos la nueva imagen del kernel:

```
root@hostname:/usr/src/linux $ make dep modules modules_install bzImage
```

Al realizar este proceso (de duración variable según la máquina que dispongamos) se ha creado una imagen del kernel que se encuentra en `/usr/src/linux/arch/i386/boot` y que se llama `bzImage`.

Copiamos esta imagen al directorio /boot y modificamos lilo para poder arrancar la imagen.

```
root@hostname:/usr/src/linux$ cp arch/i386/boot/bzImage  /boot/linux-2.4.21-adeos-m3
root@hostname:/usr/src/linux$ pico /etc/lilo.conf
```

Añadimos lo siguiente:

```
image=/boot/linux-2.4.21-adeos-m3
      label=Adeos/RTAI
      read-only

root@hostname:/usr/src/linux$ lilo
Added Linux*
Added Adeos/RTAI
```

Ahora reiniciamos el sistema y cuando salga el prompt de LiLo arrancamos la nueva imagen (Adeos/RTAI).

- Comprobación del funcionamiento de ADEOS

```
root@hostname:~$ dmesg | more
```

Deberemos ver unas líneas parecidas a estas:

```
Linux versión 2.4.21-adeos (root@hostname) (gcc versión 3.x.x) ·# 23 Mon ...
BIOS-provide physical RAM map:
....
zone(2): 0 pages
Kernel command line: auto BOOT_IMAGE=adeos ro root=302
Initializing CPU#0
ADEOS 2.4r8/x86: Root domain Linux registered.
...
```

4.1.2.- Instalación de RTAI sobre ADEOS

- Obtención de la distribución

  La versión de RTAI se puede obtener a través de http://www.aero.polimi.it/RTAI/ . Una vez localizada la guardamos en el directorio `/usr/src`.

- Instalación
  - Descomprimir la distribución

    ```
    root@hostname:~$ cd /usr/src
    root@hostname:/usr/src$ tar -xzf rtai-24.1.12.tgz
    ```

    Se ha creado un directorio llamado `rtai-24.1.12`.

  - Configuración

    ```
    root@hostname:/usr/src$ cd rtai-24.1.12
    root@hostname:/usr/src/rtai-24.1.12$ make menuconfig
    sh ./configure –menu
    Enter location of Linux source tree [/lib/modules/2.4.21-adeos/build]: <ENTER>
    ```

De todas las posibles opciones de configuración de RTAI elegiremos las siguientes inicialmente. Una vez conozcamos más RTAI es posible que necesitemos configurar otras, pero ahora dejamos por defecto el resto de opciones.

```
Code maturity level options -->
  [*] Prompt for experimental code
Test and Examples -->
  [*] Compile examples in kernel space
  [*] Compile examples in user space
  [*] Compile tests
```

- Compilado

```
root@hostname:/usr/src/rtai-24.1.12$ make dep
root@hostname:/usr/src/rtai-24.1.12$ make
root@hostname:/usr/src/rtai-24.1.12$ make install
```

Si como resultado de ejecutar este último comando sale el siguiente warning, deberemos modificar el fichero `/etc/modules.conf` añadiendo una nueva linea: `prune rtaisyms`.

```
WARNING:
You are missing the line 'prune rtaisyms' in /etc/modules.conf, which means
that each time depmod is run, it will print the warning:
  depmod: /lib/modules/2.4.21-adeos/rtaisysms is not an ELF file
```

Si nunca antes habías instalado RTAI en el sistema debes ejecutar el siguiente comando:

```
root@hostname:/usr/src/rtai-24.1.12$ make dev
```

- Comprobación del funcionamiento de RTAI sobre ADEOS

Para comprobar el funcionamiento de RTAI sobre ADEOS lo primero que hay que hacer es cargar ADEOS y después RTAI. La distribución que hemos instalado nos lo facilita en gran manera.

NOTA: Antes de cargar un módulo de kernel es conveniente sincronizar el disco para evitar perdidas de datos y cuelgues en el sistema. Se hace mediante el comando `sync`.

```
root@hostname:/usr/src/rtai-24.1.12$ sync
```

Carga de ADEOS:

```
root@hostname:/usr/src/rtai-24.1.12$ insmod modules/adeos.o
```

Carga de RTAI:

```
root@hostname:/usr/src/rtai-24.1.12$ ./ldmod
```

Para comprobar que los módulos se han cargado disponemos del comando `lsmod`, que nos debería mostrar:

```
Module           Size   Used by      Not tainted
rtai_shed        45140  0
rtai_fifos       16424  0
rtai             10796  1     [rtai_shed rtai_fifos]
adeos            11488  0     [rtai_shed rtai_fifos rtai}
```

© 2004 Diego LÓPEZ ZAMARRÓN

Y para comprobar que está en funcionamiento:

```
root@hostname:/usr/src/rtai-2.4.1.12$ dmesg
...
ADEOS: Pipeline started.
ADEOS: Domain RTAI registered.
RTAI/Adeos mounted

=== RT memory manager v1.3 Loaded. ===
***** STARTING THE UP REAL TIME SHEDULER WITH LINUX *****
***** FP SUPPORT AND READY FOR A PERIODIC TIMER *****
***<> LINUX TICK AT 100 (HZ) <>***
***<> CALIBRATED CPU FREQUENCY ___________ (HZ) <>***
***<> CALIBRATED TIMER-INTERRUPT-TO-SHEDULER LATENCY ____ (ns) <>***
***<> CALIBRATED ONE SHOT SETUP TIME ____ (ns) <>***
```

Los valores que aquí aparecen deberían corresponderse con los proporcionados por nuestro procesador. Si son totalmente erróneos simplemente no se proporcionaría tiempo real.

4.2.- Enlaces

- Documentos:
  - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/choosing_an_rtos.pdf
  - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/embedded.linux.distro.pdf
  - http://gayuba1.datsi.fi.upm.es/~dlopez/cache/doc/selection_of_rt_linux.doc

5.- Driver sobre la distribución elegida

En este punto sabemos que la distribución elegida ha sido RTAI, y el hardware del que vamos a realizar un driver es una tarjeta de 32 entradas y 32 salidas, todas ellas digitales, que se controla a través del puerto paralelo del ordenador. Esta tarjeta ha sido desarrollada por el departamento DISAM (www.disam.upm.es) de la UPM (www.upm.es) dentro del proyecto PAUTA.

Para la realización del driver y la aplicación de usuario me he servido de los siguientes libros y manuales:

- Linux Device Drivers. 2ª Edición. Ed. O'REILLY.
- RTAI Programming Guide 1.0.

Y consultando en las siguientes páginas y sus foros:

- http://www.aero.polimi.it/~rtai/index.html (Programación en RTAI)
- http://www.beyondlogic.org/ (Descripción del puerto paralelo)

5.1.- Descripción del hardware

5.1.1.- Introducción

El objetivo del módulo de entrada/salida es poder conectar un PC con diferentes tipos de dispositivos, los cuáles aceptan dos valores (activado y reposo).

Como elemento de salida primario del PC se considera el puerto paralelo o puerto de la impresora, dicho puerto se ha diseñado para suministrar datos desde el PC a otros periféricos utilizando como unidad fundamental el byte. Por lo que principalmente es un puerto de salida del PC, sin embargo posee determinadas líneas que permiten a dichos periféricos, impresoras fundamentalmente, comunicarle su estado. Estas líneas son líneas de control, y cada una de ellas tiene asociado un significado, no se tratan de líneas de datos.

Debido a que el número de líneas que el PC posee tanto de salida, como sobre todo de entrada, es limitado, se limita el número de elementos que podríamos conectar de forma directa al puerto paralelo (por ejemplo tendríamos 8 líneas de salida para el bus de datos). Para aumentar el número de elementos que se pueden controlar se recurre a la multiplexación, tanto de las entradas como de las salidas. Esto conlleva a reducir la frecuencia máxima con la que se podría cerrar un lazo de control.

El número de entradas/salidas elegido se ha fijado en 32 entradas y 32 salidas, lo que permite una frecuencia máxima de 400 Hz.

5.1.2.- Descripción del puerto paralelo

Un PC dispone de tres zonas en el espacio de direcciones de entrada/salida para controlar los puertos paralelos. Estas direcciones son:

3BC-3BF	Puerto Paralelo en la tarjeta MDA
378-37F	Primer Puerto Paralelo
278-27F	Segundo Puerto Paralelo

Por defecto se usa la zona de direcciones del primer puerto paralelo. A su vez cada zona de direcciones de un puerto paralelo dispone de tres registros (puertos físicos). En el desarrollo del driver unicamente se utilizan los registros de datos y de estado, el primero como puerto de salida, y el segundo como puerto de entrada.

- **Registro de datos (378h)**

Son las 8 líneas mediante las que se envía los datos al exterior. Son líneas de sólo escritura.

- **Registro de estado (379h)**

Son las líneas por las cuales se comunica el estado del dispositivo, normalmente la impresora, al PC. Son líneas de sólo lectura.

BIT	SIGNIFICADO
b7	Busy (Es negado internamente por la lógica del puerto)
b6	ACK
b5	PE (Sin papel)
b4	SLCT(On line)
b3	ERROR
b2	0
b1	0
b0	0

- **Registro de control (37Ah)**

Sirve para el control del hardware.

BIT	SIGNIFICADO
b7	---
b6	---
b5	---
b4	Busy
b3	ACK
b2	PE
b1	SLCT
b0	ERROR

5.1.3.- Descripción de la placa

La tarjeta es una interfaz de adaptación que permite a un PC obtener el estado de 32 entradas digitales, así como modificar el estado de 32 salidas digitales a través del puerto paralelo.

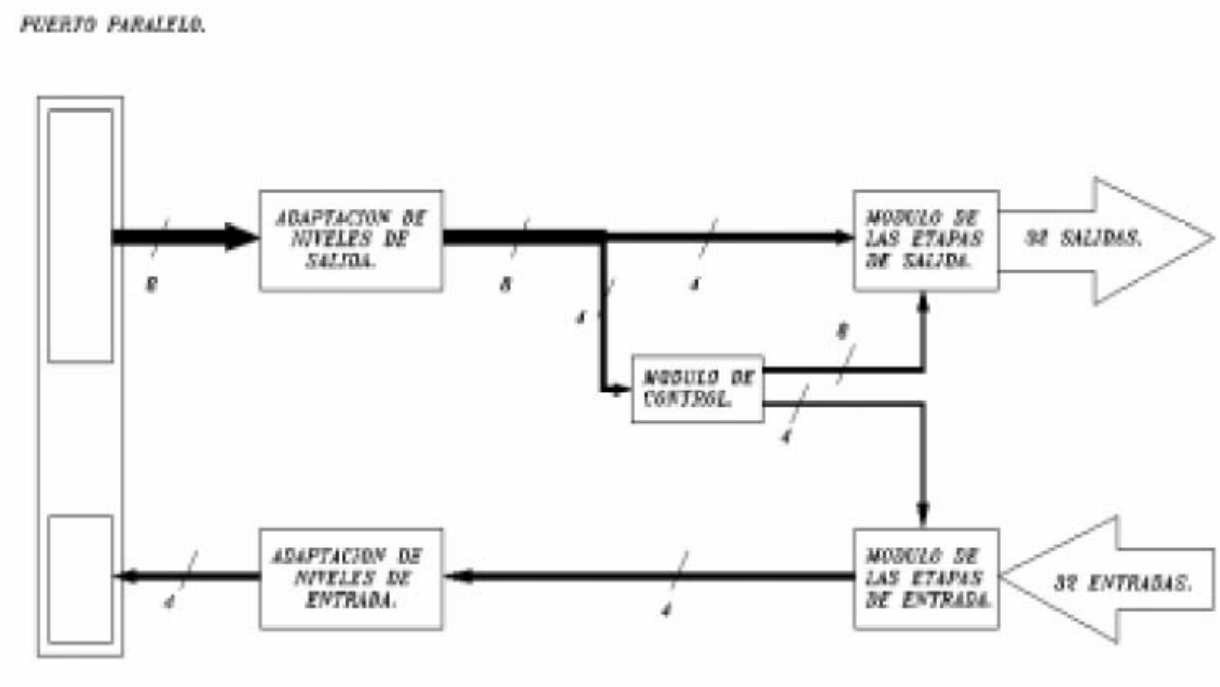
© 2004 Diego LÓPEZ ZAMARRÓN

5.1.3.1.- Descripción lógica

El diagrama de bloques de la tarjeta es el mostrado en la figura.

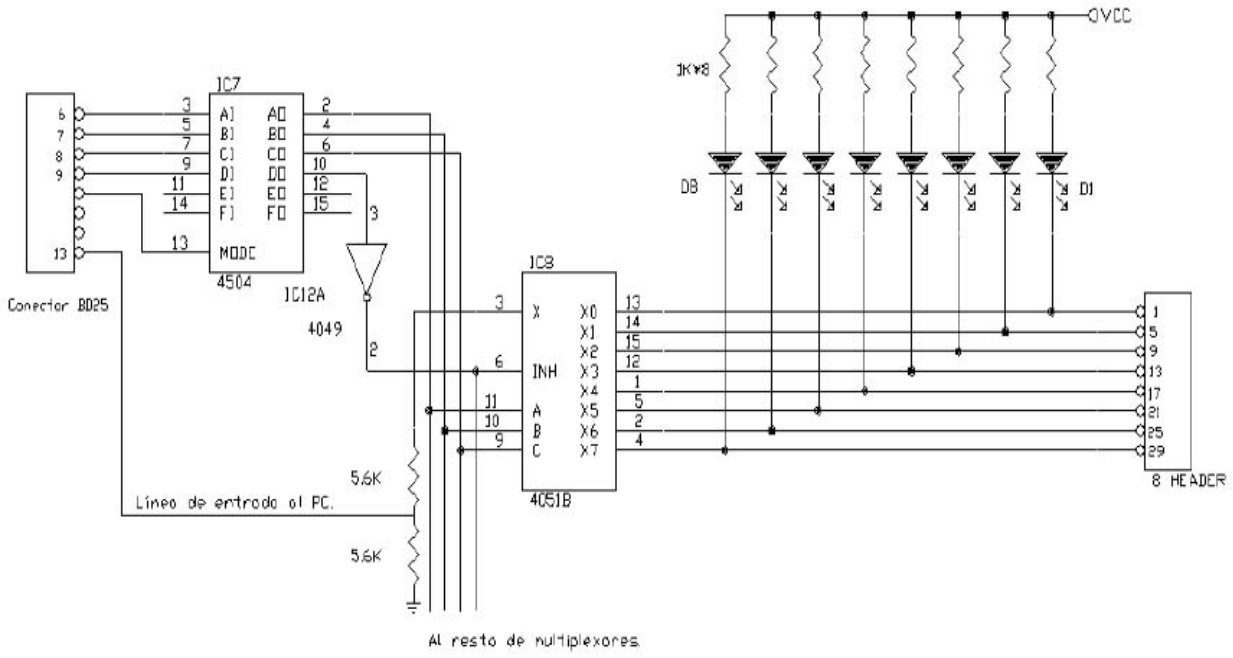
Consta de cinco bloques fundamentales:

- Adaptación de los niveles de salida.
- Adaptación de los niveles de las entradas.
- Módulo de control. Se encarga de controlar si la operación a realizar es de salida o de entrada y que grupo de salidas o entradas están activos en ese instante.
- Módulo de las etapas de salida. Se encarga de activar y visualizar las salidas correspondientes.
- Módulo de las etapas de entrada. Se encarga de visualizar y capturar el valor de las entradas correspondientes.



5.1.3.1.1.- Célula de entrada básica

La célula de entrada básica se compone de un multiplexor, y 8 leds de color rojo con sus correspondientes resistencias de polarización, tal y como se muestra en la figura.



El proceso de lectura se realiza sobre un grupo de 4 entradas (no es posible leer de forma individual) de la siguiente forma:

- En el puerto paralelo se escribe como dato un byte con el siguiente patron: b7 b6 b5 b4 b3 b2 b1 b0

Donde:

b7 = 1	Operación de lectura
b6	Selección de 1 de los 8 grupos disponibles
b5	
b4	
b3	Sin información relevante
b2	
b1	
b0	

- Al estar b7 a 1, se realiza operación de lectura por lo que se inhabilitan los latches.
- Mediante las lineas b6, b5 y b4 indicamos que grupo de 4 entradas vamos a leer. El multiplexor seleccionara entre sus 8 entradas cual será la que se conectará con la salida.
- El valor de tensión convenientemente tratado es puesto en una de las lineas de entrada del puerto paralelo.
- Desde programa se procederá a leer el registro de estado del puerto paralelo.

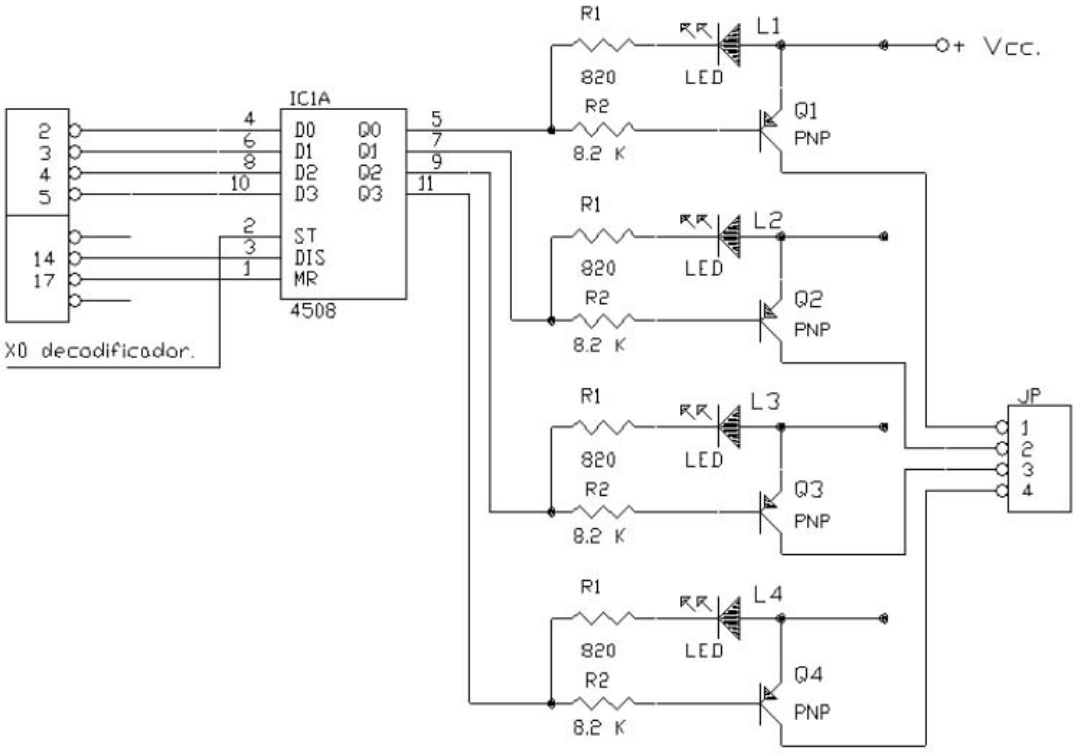
La distribución de las entradas es efectuada de forma que cada grupo de 4 sea leido a la misma vez.

© 2004 Diego LÓPEZ ZAMARRÓN

5.1.3.1.2.- Célula de salida básica

La célula de salida básica queda formada por un grupo de 4 latches, es la mitad de uno de los circuitos integrados 4508, cuatro transistores PnP con una resistencia de bas de 8,2 kilohmios y cuatro leds de color verde para indicar la activación de las salidas.

Las salidas son los colectores de los transistores tal y como se muestra en la figura.



La selección de que grupo de latches son los que están activos se realiza mediante la correspondiente decodificación, como a continuación se detalla:

- En el puerto paralelo procedemos a escribir un byte con el siguiente patrón: b7 b6 b5 b4 b3 b2 b1 b0

Donde:

b7 = 0	Operación de escritura
b6	Selección de 1 de los 8 grupos disponibles
b5	
b4	
b3	Información de los valores que queremos escribir en el grupo
b2	
b1	
b0	

- Al estar b7 a 0 e inhabilitan las operaciones de lectura y se activa el decodificador con el que seleccionamos que grupo de cuatro latches se activa. Con la información presente en b6, b5 y b4 seleccionamos que grupo de cuatro latches será direccionado.
- La información presente en b3, b2, b1 y b0 es pasada a los latches, después de ser invertida, esto se realiza para trabajar con lógica positiva en las salidas, escribiendo un 1 en la correspondiente linea del puerto paralelo, al invertirlo se obtiene un nivel bajo, el cual tras ser guardado por el latch, produce la conducción del transistor correspondiente, con lo que en el colector tendremos un nivel alto de tensión, es decir un 1, tal y como se escribió.
- La decodificación y multiplexación provoca que se actualizen cuatro líneas simultaneamente, los grupos son de cuatro en cuatro y correlativos.

Hay que tener en cuenta que se está trabajando con latches, y no con biestables gobernados por reloj, esto puede provocar que en cierto equipos cuya rapidez sea elevada, se produzca la captura por el grupo de latches anterior del valor que deseamos escribir en el siguiente grupo de latches. Para correguir este efecto se deben realizar dos escrituras, una en el grupo nuevo con el valor viejo, y otra, en el grupo nuevo con el valor nuevo.

Otra cosa a tener en cuenta es el orden de los grupos. Su orden lógico no coincide con el físico, el cuál sigue una disposición de codigo Gray.

5.1.3.1.3.- Niveles lógicos de las entradas

El dispositivo de entrada/salida puede suministrar directamente la polarización de las entradas, de tal forma que si se dispone un interruptor o pulsador entre una entrada o masa se tendrá:

- Pulsador en circuito abierto --> Entrada en nivel Alto.
- Pulsador en cortocircuito -- > Entrada en nivel bajo.

Se propone de todas formas disponer de las resistencias correspondientes en los pulsadores o interruptores de tal forma que se puedan fijar un nivel alto o bajo con independencia del dispositivo de entrada.

5.1.3.1.4.- Niveles lógicos de las salidas

Tal y como se ha comentado previamente un nivel alto en la correspondiente línea del puerto paralelo, se traduce en conducción del transistor correspondiente y por tanto en un nivel alto en la salida pertinente. Obsérvese que se dispone el nivel alto mediante la conducción de un transistor, si en la salida se presenta un cortocircuito o una resistencia muy baja, se puede llegar a provocar la ruptura del transistor. Las intensidades por cada una de las salidas no deben superar los 100mA.

5.1.3.2.- Descripción física

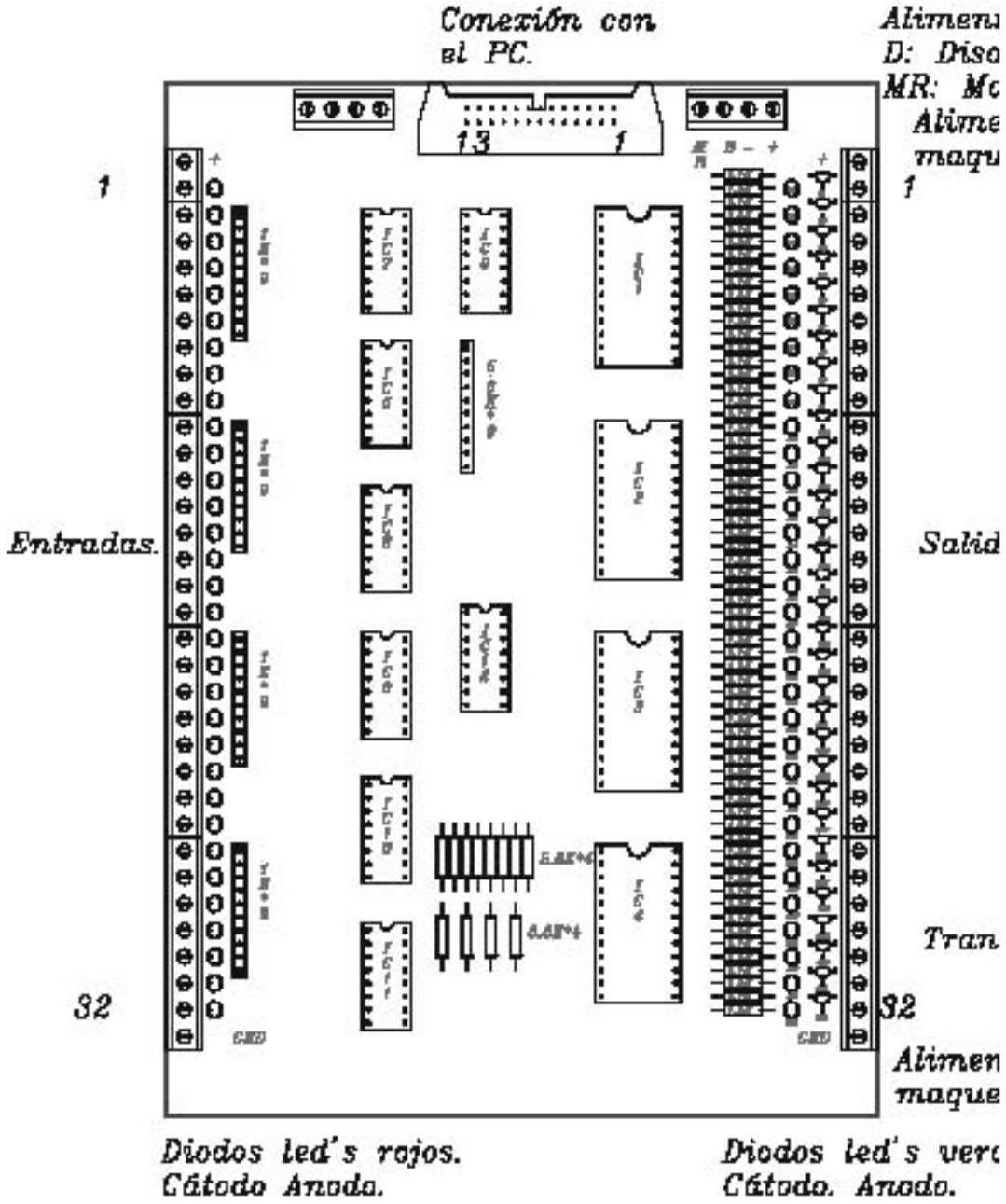
5.1.3.2.1.- Tensiones de alimentación

Como anteriormente se ha mencionado se utilizan circuitos integrados fabricados con tecnología CMOS, esto permite una amplia variación en la tensión de alimentación, desde aproximadamente 15 V hasta unos 3 V, sin embargo la tensión con la que se recomienda trabajar es de 9 V, suministrada a través de una pila, el cálculo de los valores de las resistencias de polarización han sido realizados para este valor de tensión de alimentación.

© 2004 Diego LÓPEZ ZAMARRÓN

5.1.3.2.2.-Identificación de terminales

En la figura se pueden observar las conexiones y componentes.



5.1.3.2.3.- Disposición de los circuitos integrados

En la figura anterior se puede apreciar la localización de los circuitos integrados, así como su disposición, en la parte izquierda están presentes los multiplexores mientras que en la parte derecha se disponen los latches. En la parte superior, junto al conector de cable plano, están presentes los adaptadores de nivel.

Es preciso tener presente la presencia de la muesca para no insertar los integrados de forma errónea, en todos ellos la patilla número 1 es la superior izquierda.

5.1.3.2.4.- Lista de material

Lista de material.

Módulo de Entrada / Salida.

Dispositivo.	Denominación	Valor.	Nº unidades.
Placa de circuito impreso	Doble cara.	(198mm * 148 mm)	1
Clemas de 2 contactos			2
Clemas de 4 contactos			2
Clemas de 8 contactos			8
Conector puerto paralelo	DB25		1
Cable conexión puerto paralelo			1
Latches	IC1 - IC4	4508	4
Multiplexores/Demultiplexores	IC5; IC8 - IC11	4051	4
Adaptadores de nivel TTL /CMOS	IC6- IC7	4504	2
Inversores	IC12	4009	1
Array de resistencia		1 K * 9	4
Array de resistencia		5.6 K * 9	4
Resistenicas		5.6 K	8
Diodos		1N-----	4
Resistenicas		9.1 K (1/8W)	32
Resistenicas		1,2 K (1/8W)	32
Led rojo	D1 - D32	Diámetro 5mm	32
Led verde	L1 - L32	Diámetro 5mm	32
Transistores(PNP)	Q1 - Q32	2N-----	32
Tornillería.(M3)			

5.2.- Descripción del software

Este trabajo no pretende realizar un manual de como realizar drivers, módulos, ni aplicaciones para RTAI o Linux, simplemente se describirá el diseño escogido para realizar el driver del hardware anteriormente descrito, junto con una librería de uso y una aplicacion de usuario de ejemplo. El diseño escogido no es el único y quizás no sea el mejor, simplemente es uno que funciona bien, que por suspuesto siempre es mejorable, y que invito al lector a mejorar en cualquier aspecto que considere conveniente.

© 2004 Diego LÓPEZ ZAMARRÓN

5.2.1.- Driver

A la hora de realizar el driver hemos tenido en cuenta los consejos del libro “Linux Device Drivers 2ª Edición”. Uno de estos consejos evidentemente era que necesitabamos programar al menos un módulo, con todos los problemas que conlleva, pero también con todas las ventajas que nos proporciona acceder, por ejemplo, directamente a las posiciones de memoria del kernel, o a las funciones definidas por éste.

Teniendo en cuenta todas las ventajas y desventajas que aporta el uso de módulos para el desarrollo de drivers, lo primero que debemos tener en cuenta es lo que podemos hacer con el hardware, para saber que es lo que debemos programar en el driver. En este caso el hardware es bien sencillo, escribir o leer por el puerto paralelo el valor de las entradas o salidas digitales de que dispone. El pequeño inconveniente que tenemos es que a través del puerto paralelo tan sólo podriamos controlar 8 entradas o salidas simultáneamente, por lo que se recurre a la multiplexación. Aqui, en este pequeño problema, es donde radica el uso de un driver especial y distinto del típico driver para el puerto paralelo, porque ahora el nuevo driver debe implementar una lógica de uso distinta de la habitual, porque la parte física no cambia, el puerto paralelo tiene sus restricciones de uso (registro de 1 byte de tamaño).

Una vez sabemos lo que tenemos que implementar planteamos el módulo. En este caso el módulo se divide en dos ficheros:

- Fichero de definición de cabeceras (rtai_ensa.h), donde se definen cabeceras de funciones, estructuras de datos y constantes.
- Fichero de código (rtai_ensa.c), donde se implementan las funciones. Las funciones que se implementan son las básicas para crear un módulo, que son:
  - int init_module(void): Función que inicializa el módulo en el kernel. Chequea/registra el espacio de memoria necesario e inicializa el driver.
  - void cleanup_module(void): Función que descarga el módulo del kernel. Libera la región de memoria usada por el driver.

  Y las específicas del driver que son:
  - ssize_t rtaiensa_read(long *data_out): Función que lee 32 entradas digitales desde el puerto paralelo.
  - ssize_t rtaiensa_write(long data_in): Función que escribe 32 salidas digitales por el puerto paralelo.
  - int rtaiensa_open(void): Función que abre el dispositivo. Incrementa el contador de uso del driver.
  - int rtaiensa_close(void): Función que cierra el dispositivo. Decrementa el contador de uso del driver.

*Nota: Destacar el uso del tipo de datos “long” en las funciones rtaiensa_read y rtaiensa_write, que en la arquitectura x86 proporciona un tamaño de 32 bits (1 bit por entrada/salida), pero que impide que el driver sea multiplataforma. Como ya se comento anteriormente, se invita al lector a la modificación.

*Nota: En las funciones rtaiensa_read y rtaiensa_write es donde se ha implementado la lógica necesaria para este dispositivo, es decir, la escritura “simultanea” sobre las 32 entradas/salidas a través del puerto paralelo.

En este caso, la arquitectura que se ha seguido para desarrollar el driver no permite el uso de estas funciones directamente por aplicaciones de nivel de usuario. Ésto no tiene gran importancia ya que el driver que se ha realizado es para tiempo real, y las mejores prestaciones se obtienen en el nivel de nucleo. Aún así, es posible comunicarse con aplicaciones de nivel de usuario, pero esto se comentará en los siguientes puntos.

© 2004 Diego LÓPEZ ZAMARRÓN

5.2.2.- Librería

Como se ha comentado en el punto anterior, el diseño elegido no permite el uso del driver directamente desde el nivel de usuario, así que se ha creado una librería de uso. La librería es un módulo también (librtaiensa.c).

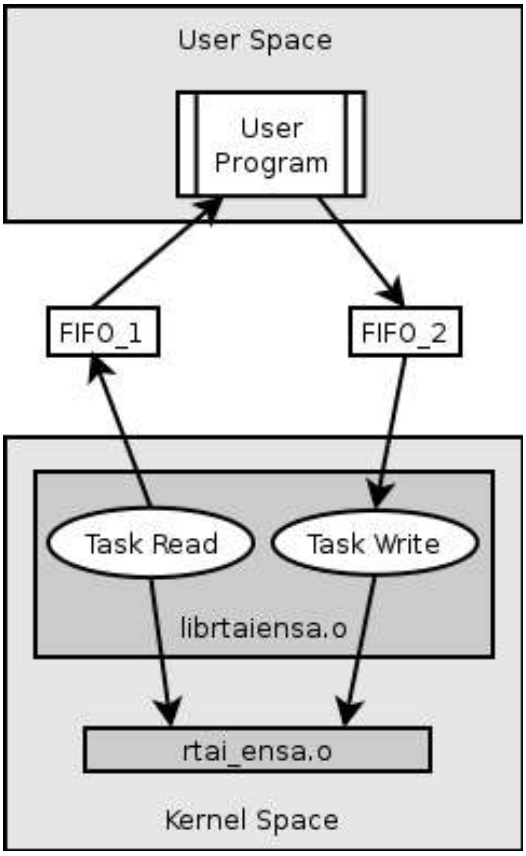
Este módulo implementa 2 tareas de tiempo real, una que se encarga de leer periodicamente, y otra en escribir, de la tarjeta de entrada/salida.

La tarea de lectura, lee del dispositivo haciendo uso de la función proporcionada por el driver y escribe el valor obtenido en un FIFO para que la aplicación de usuario pueda obtener el valor. A continuación espera a que se cumpla el periodo establecido y vuelve a leer, asi de forma indefinida.

La tarea de escritura, escribe en el dispositivo(usando el driver) cuando la aplicación de usuario proporciona un valor a través de otro FIFO.

Dado que en el diseño del driver no se ha tenido en cuenta su uso de forma concurrente por varias aplicaciones, se ha tenido que definir un semaforo para conseguir acceso exclusivo al dispositivo y evitar lecturas/escrituras simultaneas sobre el dispositivo, cosa que éste no soporta.

Puesto que la librería es un módulo son necesarias dos funciones, init_module(), donde se crean los fifos y el semaforo y se arrancan las tareas, y cleanup_module(), donde se borran los fifos y el semaforo, y se borran las tareas. Además en el init_module() se inicializa un manejador para el fifo que usa la tarea de escribir. Éste manejador despierta a la tarea escritora cuando se ha introducido un dato en el fifo por parte de la aplicación de usuario.



5.2.3.- Aplicación

La aplicación de usuario es muy sencilla con una interfaz tipo texto lo suficientemente clara para hacer pruebas y poder actuar sobre todas las entradas y salidas, tanto de forma conjunta, como de forma individual.

Lo más destacable de la aplicación es el uso, tambien aqui, de dos fifos, uno de escritura (la librería lee por este fifo), y otro de lectura (la librería escribe por este fifo), que permite la comunicación con la libreria situada en el nivel del kernel.

© 2004 Diego LÓPEZ ZAMARRÓN

Anexo 1: GNU Free Documentation License

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and estandard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-estandard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a estandard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document
> under the terms of the GNU Free Documentation License, Version 1.2
> or any later version published by the Free Software Foundation;
> with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
> A copy of the license is included in the section entitled "GNU
> Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the
> Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.